UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO DE FINAL DE GRADO

**Grado en Ingeniería de Materiales.**

# PROYECTO SPAGHETTI

## Memoria y Anexos


**Autor/a:** Daniel Gutiérrez Sánchez
**Director/a:** Emilio Jiménez
**Co-Director/a**: Kim Albó

**Convocatoria:** Junio 2020

# Resumen

Las propiedades mecánicas de los materiales se determinan realizando ensayos en laboratorio que reproduciendo las condiciones de servicio. Para determinarlos es necesario estudiar la naturaleza de la carga aplicada, su duración, y las condiciones del medio. En el caso de la carga puede ser a tracción, a compresión o a cizalladura, además su magnitud puede ser constante o bien fluctuar continuamente. Sobre la duración pueden ser ensayos de días y meses a segundos. Por último las condiciones del medio pueden depender de la temperatura de servicio y de la humedad entre otros.

Este Proyecto de Final de Grado nace de un anterior en la materia de Proyectos de Ingeniería de Materiales. Donde se propuso realizar una máquina de ensayo universal de materiales blandos a Tracción. Actualmente no existe máquina con estas características en el mercado, por lo que se decidió a construirla de cero.

Como objetivo se realizará una mejora general en el funcionamiento de la máquina universal de ensayo propuesta en la asignatura de Proyectos de Ingeniería de Materiales. La creación de un nuevo Software intuitivo que permita realizar diferentes tipos de ensayos: a tracción, a compresión, fatiga; con diferentes tipos de controles e incluso especializados como podría ser un ensayo Creep. Este nuevo Software se diseñará para utilizarse en la máquina ya creada, pero con la idea de poder ser reutilizado en otras máquinas de la Facultad.

# Resum

Les propietats mecániques dels materiales es determinen realitzant assaigs a laboratoris reproduint les condicions en servei. Per determinarles es necesari estudiar: la naturalesa de la carrega aplicada, la seva duració i les condicions del medi. En el cas de la carréga tenim assaigs a tracció, a compresió i a cizalla. Parlant sobre la duració poden ser de segons o de mesos i dies. Per últim, les condicions del medi que poden afectar a les propietats mecániques poden ser temperatura o humitat entre altres.

Aquest projecte de Fi de Grau naix d'un anterior a l'assignatura Projectes d'Enginyeria de Materials. On es va proposar realitzar de cero una màquina de assaig universal de materials tous a Tracció o compressió. Actualment no existeix una màquina amb aquestes caraterístiques al mercat.

Com a objetctiu es realitzara una millora general en el funcionament de la màquina universal d'assaig a l'assignatura de Projectes d'Enginyeria de Materials. La creació d'un nou Software intuitiu que sigui capaç de realitzar diferents tipus d'assaig: a tracció, compressió, fatiga; Amb diferents tipus de controls i especailitzats com podrien ser assaigs de Creep. Aquest nou Software es disenyarà per poder ser utilitzat a la màquina ja creada, però amb la idea de poder-se instalar en qualsevol màquina de la facultat.

# Abstract

The mechanical propierties of materials are defined by testing in laboratory reproducing the service conditions. To determine the mechanical propierties is necesary to study: the nature of the applied load, the load duration and the environmental conditions. Studying the load there are tensile test, compression or shear. The duration can be seconds or either months. Finally, there are some environmental conditions can affect directly to mechanical propierties like temperature and humity.

This project was born in a previous subject called Material Engineering Projects. Where was propoused to make a universal test machine from zero for testing soft materials. Actually, there not precedents so it would be the first one in the market.

The objective of this project is improve the machine and his functionallity by creating a new intuitive Software capable of doing tensile or compression tests, also fatigue test or other types of testings like Creep. This new Software it will be designed for being used in the machine made before, but changing some inputs and outputs it will be allowed to work in any kind of machine.

# Agradecimientos

Primeramente, me gustaría agradecer a ambos tutores, Emilio Jiménez y Kim Albó, por depositar la confianza en mí para realizar este proyecto habiendo cursado la anterior asignatura con ambos.

Agradecer sobre todo a Kim Albó su paciencia y su ayuda constante para formarme en LabView, y guiarme en este proyecto que era totalmente nuevo para mí.

Además, agradecer a mis compañeros de clase que realizaron conjuntamente la máquina inicial.

# Índice

# Prefacio

## 1.1 Origen del trabajo

Durante el cuadrimestre de otoño del curso 2019-2020, en la asignatura de Proyectos de Ingeniería de Materiales se presentó el *Proyecto Spaghetti*, un proyecto que trataba de realizar desde cero, una máquina de ensayos para analizar las propiedades mecánicas de la pasta.

El proyecto se realizó con éxito, pero quedando la posibilidad de realizar mejoras en el software como en el hardware, además de su utilización para aplicaciones ingenieriles.

## 1.2 Motivación

Actualmente en el mercado no existe una máquina de tracción / compresión de pequeñas dimensiones, por lo que la construcción de una podría permitir analizar y estudiar diferentes materiales frágiles o sencillamente de dimensiones más reducidas.

Además, comprender el funcionamiento general de las máquinas de ensayo, como se genera el par de fuerza, como se convierte la magnitud física en eléctrica y la teoría existente detrás de ella.

En lo personal la motivación de realizar este proyecto reside en diversos puntos:

- El primero fue la sensación que el proyecto de la materia pasada, se quedó a medio hacer. Se consiguió que la máquina hiciera su función, pero no de manera óptima. Por lo tanto, el profundizar, añadir un interfaz de software como LabView, permitiendo así obtener gráficas y analizar *in-situ* los resultados, y poder optimizar su funcionamiento resultaba muy de mi agrado.
- En segundo lugar, es la experiencia que puede aportar aprender a utilizar dos de los software más populares y usados de la actualidad, como son Arduino y LabView.
- Por último, la posibilidad de que máquinas con un software ya obsoleto puedan tener una "segunda vida útil" gracias a el programario que se ha creado en este proyecto.

# Introducción

## 1.3 Objetivos del trabajo

En este proyecto se destacan dos grandes objetivos:

El principal objetivo es aprender a realizar una interfaz intuitiva y atractiva para el usuario, con LabView, que permita ser instalado en cualquier máquina de ensayo universal.

El segundo objetivo de este trabajo de fin de carrera es conseguir que la máquina de ensayos obtenga correctamente valores de fuerza y desplazamiento, para así poder almacenar datos y crear gráficas que permitirán el estudio de los materiales ensayados.

## 1.4 Alcance del trabajo

En este trabajo, para conseguir los objetivos previamente mencionados, se debe primero familiarizarse con LabView. Entender y conseguir un nivel de programación adecuado para proceder a la creación del software. Este software deberá ser capaz de interpretar las señales eléctricas que recibe tanto de la célula de carga como del encoder, para poder transformarlas en valores de fuerza y desplazamiento.

Para conseguir que la célula de carga interprete correctamente los valores de fuerza, se deberá calibrar anteriormente mediante un dinamómetro y así calibrarla.

Esta máquina puede servir para realizar diferentes estudios, por ejemplo, poder analizar las propiedades mecánicas de bolsas de compra, y compararlas con su impacto medioambiental.

# 2. Marco teórico.

## 2.1 Tipos de Ensayos.

Mediante un ensayo de esfuerzo-deformación se puede conocer y estimar el comportamiento mecánico de una pieza que está sometida a una carga estática o dinámica en sobre una sección o superficie.

Principalmente existen tres tipos de ensayo a tracción, a compresión y a cizalladura, aunque también existen flexión y torsión, mostrados en la figura 1. En este proyecto únicamente se va a utilizar el ensayo a tracción y a compresión, por lo que el ensayo a cizalladura será omitido [1]. Aunque cambiando las mordazas de la máquina se podría llegar a estudiar ensayos de flexión a diferentes puntos.

*Figura 1. De izquierda a derecha, tracción, compresión, flexión, torsión y cizalladura.*

### 2.1.1 Ensayo a Tracción.

Es el ensayo más común a realizar. Este puede ser utilizado para determinar varias propiedades de los materiales que son importantes para el diseño. Para realizar este estudio normalmente se deforma una probeta hasta la rotura, con una carga de tracción aplicada uniaxialmente en el eje de la probeta. Hay diferentes tipos de probetas, pero todas están sujetas a las normas DIN 53455, ISO 6892-1 y la ASTM E8/8m [2] que especifica su longitud, su sección constante y su anchura, todas las dimensiones se muestran en la figura 2. Además, esta norma determina como se debe realizar el ensayo. Este ensayo, como los otros tres, es destructivo, por lo que la probeta o es deformada permanentemente o rota. [1]

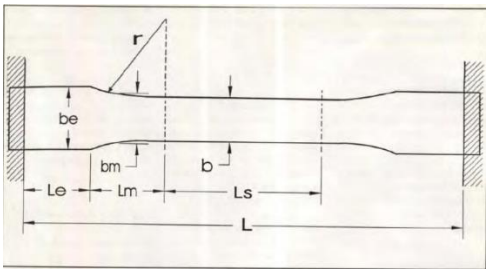



*Figura 2. Probeta según norma DIN 53455.*

Para obtener las características mecánicas de un material se realizan gráficas Tensión-Deformación. Estas graficas se construyen a partir de las ecuaciones 1 y 2. En la ecuación 1, se obtiene que la tensión (expresada como sigma(σ)) es el resultado de dividir la fuerza obtenida por la célula de carga, entre el área perpendicular a el eje de la fuerza, por lo tanto, sus unidades son los MPa ($10^6$ N/$m^2$). En cambio, la deformación consiste en el incremento de la longitud divido entre la longitud inicial. Este no tiene unidades, aunque se puede expresar de 3 maneras diferentes, o en tanto por uno, en tanto por ciento o mm/mm. [1]

*Ecuación 1. Tensión.*

$$\sigma = \frac{F}{Ao}$$

*Ecuación 2. Deformación.*

$$\varepsilon = \frac{li - lo}{lo} = \frac{\Delta l}{lo}$$

Gracias a la normalización de la probeta, se pueden comparar los valores de diferentes experimentos, ya que una diferente sección provocaría una tensión diferente.

### 2.1.2 Ensayo a Compresión.

Los ensayos de compresión son utilizados sobre todo para elementos estructurales, por lo que son bajo deformaciones permanentes grandes. Estos ensayos no dan mucha información y se realizan igual que los ensayos a tracción. En este caso tanto la Fuerza como el incremento de longitud de la probeta serán negativos.

### 2.1.3 Tipos de control.

Para los tipos de ensayos explicados anteriormente, principalmente se controlan de dos maneras a velocidad constante o a fuerza constante:

- **Control de desplazamiento.** Se define una velocidad constante de ensayo en mm/s.

- **Control por fuerza.** Se define la velocidad constante de ensayo en N/s.

### 2.1.4 Fatiga.

La fatiga es una forma de rotura que aparece en cualquier tipo de material (metales, cerámicos, polímeros y compuestos) sometido a tensiones dinámicas y fluctuantes. Tal y como indica su nombre,

este tipo de fractura ocurre después de un período largo de tensiones reiterativas o de deformaciones cíclicas. Esta rotura, se produce incluso en tensiones menores que la resistencia a tracción o límite elástico de una carga estática, debido a la iniciación y propagación de microfisuras.

Es importante estudiar los fenómenos de fatiga, ya que se estima que, el 90% de las roturas, vienen producidas por este efecto. Por eso, es necesario simular el ensayo de tal manera que se reproduzca las condiciones en servicio. [1]

#### 2.1.4.1 Tensiones cíclicas.

La tensión aplicada puede ser axial (tensión-compresión), de flexión o de torsión, aunque en este proyecto se realizará únicamente de manera axial. En este caso existen dios tipos de tensiones, la tensión máxima o tracción ($\sigma_{máx}$) y la tensión mínima o de compresión($\sigma_{min}$), por convención el valor de estos esfuerzos positivos en el caso de tracción y negativos en el caso de compresión.

Para realizar un ensayo de fatiga es necesario conocer los siguientes parámetros que nos ayudarán a definir el ensayo:

1. El valor de medio, promedio de tensiones máximas y mínimas.

*Ecuación 3. Valor medio tensiones, fatiga.*

$$\sigma = \frac{\sigma_{máx} + \sigma_{min}}{2}$$

2. El intervalo de tensión.

*Ecuación 4. Intervalo tensiones, fatiga.*

$$\sigma = \sigma_{máx} - \sigma_{min}$$

3. La amplitud de tensión, que es la mitad del intervalo de tensiones.

*Ecuación 5. Amplitud de tensiones, fatiga.*

$$\sigma = \frac{\sigma_{máx} - \sigma_{min}}{2}$$

4. Coeficiente de tensiones.

*Ecuación 6. Coeficiente de tensiones.*

$$R = \frac{\sigma_{min}}{\sigma_{máx}}$$

El resultado de graficar la tensión respecto al tiempo tendrá estos tipos de formas recogidas por la ASTM en la figura 3. Además, la frecuencia también será un parámetro a tener en cuenta, definido por el número de repeticiones por unidad de tiempo.

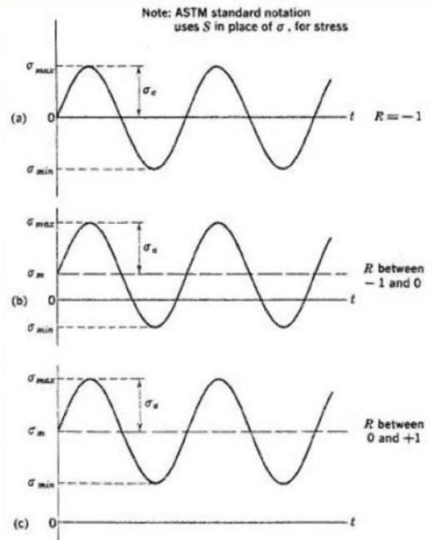



*Figura 3. Tensiones vs. tiempo en fatiga ASTM.*

#### 2.1.4.2 Curva S-N

Igual que en los ensayos de tracción se puede obtener las características mecánicas de un material, en fatiga mediante las curvas S-N se pueden obtener estas características, tienen la forma característica que se muestra en la figura 4.

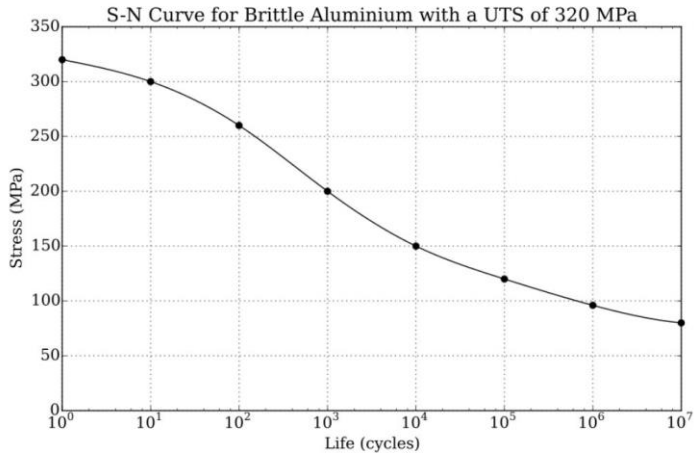



*Figura 4. Curva S-N de un Aluminio frágil.*

Las características más importantes de este tipo de curvas son:

1. La resistencia a fatiga, que se define como el nivel de tensión en el que se produce la rotura en un determinado número de ciclos. Por ejemplo, en la figura 4 a $10^4$ ciclos corresponden a 150 MPa de resistencia.

2. La vida a fatiga, que es exactamente al revés, cuantos ciclos resistirá a un nivel específico de tensión.

## 2.2 Ensayo a tracción mediante norma ASTM D638.

Para el ensayo a tracción se propone el procedimiento seguido en la Norma ASTM D638, ya que su realización permitirá poder analizar y comparar con resultados teóricos o de multitud de ensayos ya realizados [2].

- Indicar las dimensiones de la probeta de ancho y espesor.
- Verificar la sujeción de las mordazas, evitar deslizamientos.
- Fijar la velocidad, existen tablas dónde se recomiendan según el material a ensayar.
- Comprobar las condiciones de temperatura y humedad del laboratorio. Aproximadamente unos 23 ±2 ºC y 50±5% de humedad relativa.
- Se deberán ensayar 5 probetas en el caso de materiales isotrópicos y 10 en anisotrópicos.
- Se debe registrar la carga y la deformación tantas veces como sea posible.
- Se deberá tener en cuenta el deslizamiento de la máquina y otras magnitudes para evitar la toma errónea de datos.

## 2.3 Máquina universal de ensayos.

Entre las diferentes funciones que tiene una máquina universal de ensayos se encuentran:

1. Determinar las propiedades ingenieriles de los materiales y fijar las condiciones de trabajos a las cuales pueden estar sometidos.
2. Efectuar controles de calidad durante procesos de fabricación.
3. Determinar la aplicación de diferentes tratamientos térmicos.
4. Establecer causas de fallo.
5. Desarrollar nuevas aplicaciones o nuevos materiales.

Los tres principales ensayos que se realizan en una máquina universal de ensayo son tracción, compresión y flexión.

Una máquina universal de ensayo debe contener: dos cabezales, y al menos uno de ellos móviles. También se necesita un mecanismo para aplicar carga, existen máquinas servo hidráulicas y electromecánicas, una célula de carga para poder medir la fuerza aplicada, una interfaz de control con el que el usuario podrá definir variables externas y obtener información, opcionalmente se puede utilizar la ayuda de un extensómetro para medir la deformación del material ensayado, en nuestro

caso se utilizará un encoder que no es tan preciso para la deformación, pero en cambio es muy preciso para el desplazamiento, por eso mayoritariamente se realizarán preferiblemente ensayos fuerza vs.. desplazamiento.

# 3. Estado del Arte.

## 3.1 Diseño, construcción y funcionamiento.

Para el diseño de la máquina universal se han tenido en cuenta los modelos que existen en la facultad y los que hay actualmente en el mercado. Se ha observado el tipo de mecanismo y accionamiento, y se ha adaptado a unas dimensiones más pequeñas. Se puede observar en la figura 5 una máquina universal de ensayos.

*Figura 5. Máquina universal de ensayos.*

Según su estructura y su tipo de accionamiento encontramos de diferentes tipos.

### 3.1.1 Según estructura.

Existen las **monoespacio**, figura 6, esta posee dos placas enlazadas mediante dos columnas. En estas dos placas se ensamblan la electrónica necesaria y además de mordazas o puntos de apoyo, necesarios para fijar el material de ensayo. [10]

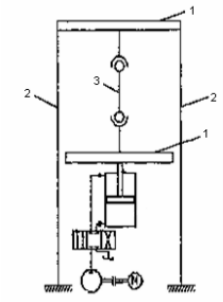



*Figura 6. Máquina universal monoespacio.*

También podemos encontrar las de **doble espacio**, figura 7. Estas disponen de dos espacios dónde realizar tracción y compresión. Consisten en 3 placas unidas por cuatro columnas. Existe una placa que comparte espacio con ambos ensayos y esta placa será la que realice el mecanismo de deformación, estirando en el ensayo de tracción y comprimiendo en el de compresión. [10]

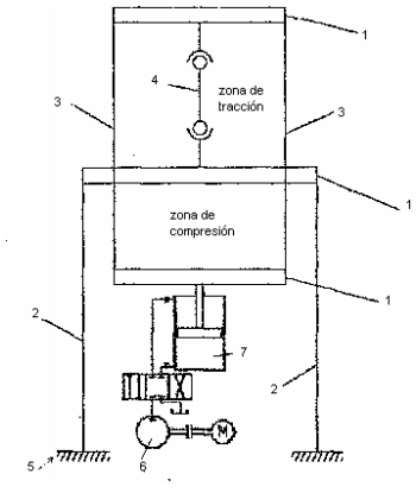



*Figura 7. Máquina universal de ensayo doble espacio.*

### 3.1.2 Según accionamiento.

**Accionamiento Hidráulico.** En la figura 7 podemos observar que es el mecanismo que utiliza para mover la placa. Se observa que esa fuerza es realizada mediante un sistema de bomba hidráulica y un actuador de doble efecto.

**Accionamiento Mecánico.** Este se realiza mediante un motor, husillos sin fin y otros tipos de transmisión del movimiento como pueden ser engranajes y poleas para aplicar la fuerza. [10]

#### 3.1.2.1 Accionamiento hidráulico.

El accionamiento hidráulico se consigue mediante la presión de un fluido una transmisión de movimiento. Se caracteriza por:

- Tener alta precisión.
- Una alta repetitividad, es decir que, en ensayos diferentes, con las mismas características presentan una muy pequeña variación entre valores.
- Buena respuesta a altas frecuencias.
- Cambios rápidos, aunque sean cambios pequeños.
- Coste elevado.

En la figura 8, se puede observar un esquema de una válvula servo hidráulica.

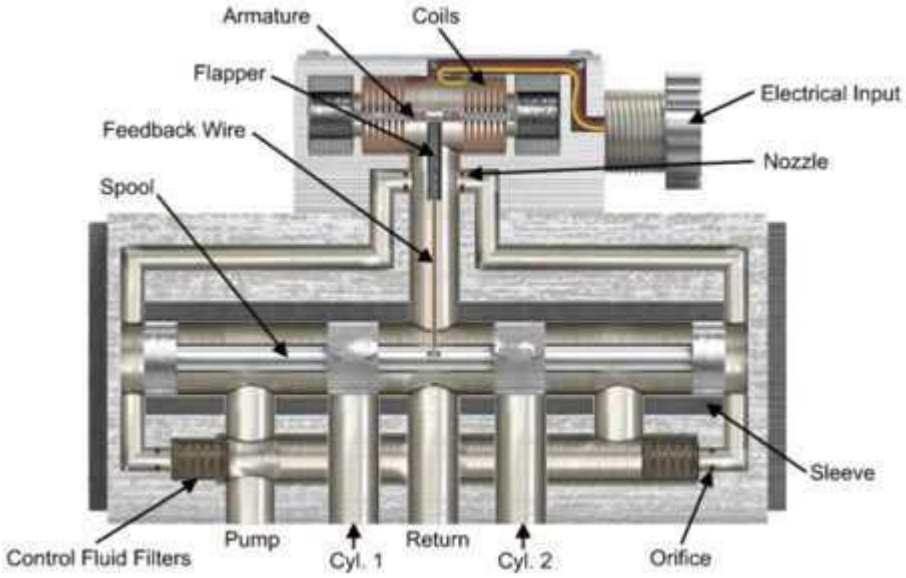



*Figura 8.Electroválvula servo hidráulica.*

Para entender cómo funciona este accionamiento primero se debe entender movimiento del fluido dentro del circuito. El recorrido se puede observar en la figura 9 como empieza por "Pump" y se desplaza hacia los dos "Control Fluid Filtrers". El fluido va inundando todo el túnel hasta llegar al "Noozle", donde se introduce hasta "Return". [15]

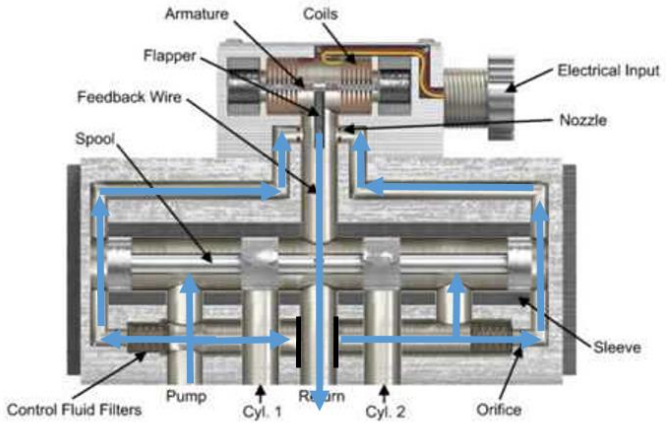



*Figura 9. Recorrido Fluido.*

En la posición de reposo, sin alimentación en las bobinas, el "flapper" estará centrado dejando libres ambas salidas "Nozzle", por tanto la presión de ambos extremos de la corredera será igual, lo que provocará que se quede en el centro tapando los dos orificios conectados a el actuador, manteniendo

éste en reposo. En el momento que una señal eléctrica excite las bobinas (en las figuras 8,9 y 10 se pueden observar como “Coils”), el torque motor moverá el “flapper” para tapar uno de los orificios. Este movimiento provocará que el aumento de presión en uno de los extremos de la corredera provocando su movimiento hacía el lado contrario, por lo que provocará que ambos orificios que permiten el movimiento del pistón de doble efecto conectado (“cyl.1 y 2”) se abran. Esto sucederá hasta que el resorte de retroalimentación tenga tanta fuerza como la aplicada mediante el campo magnético de la bobina. En este momento el orificio se empezará a destapar y la corredera en una posición de equilibro entre la fuerza del campo eléctrico creado por las bobinas, la presión en el extremo de la corredera y la fuerza aplicada por el resorte de retroalimentación.

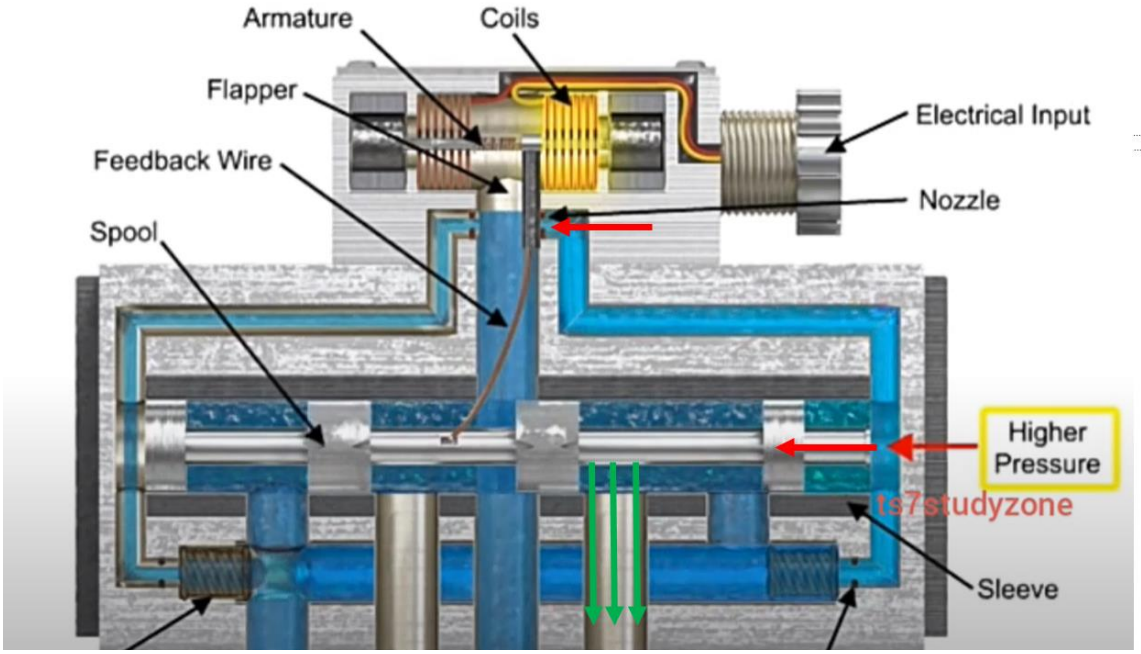



*Figura 10. Movimiento provocado por las bobinas.*

Por lo que un accionamiento hidráulico en una máquina de ensayo universal se vería tal como la figura 11. Donde mediante un ordenador, se da la orden de excitar las bobinas mediante el acondicionador, provocando una diferencia de presión, que se transmite al pistón de doble efecto, este se llama de doble efecto porque tiene dos orificios donde entra el fluido correspondiente. Este pistón ejercerá la fuerza al material a ensayar, representado en la figura como un muelle y un amortiguador. [13] [14]

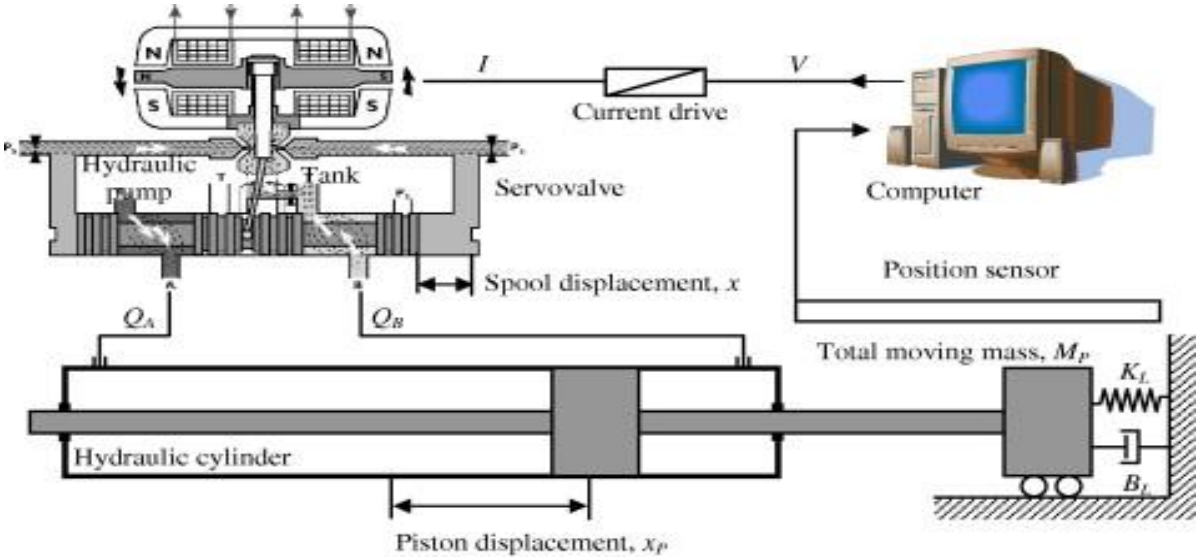



*Figura 11. Esquema máquina de ensayo universal servo hidráulica.*

#### 3.1.2.2 Accionamiento mecánico.

Para el accionamiento mecánico suele utilizarse un brushless DC motor que suele trabajar a ±80V 15A. Este motor es gobernado por un servomotor amplifier que de una señal analógica de entrada de ±10V da una salida de potencia (output power) en forma de PWM proporcional a la consigna recibida por el controlador. [15] [16] [17]

El PWM funciona a partir de porcentajes, llamados ciclos de trabajo, y la señal entrante, por ejemplo, una señal de ±10V en el analog input de un servomotor amplifier. En la figura 12 se pueden observar el efecto que producen. Un porcentaje de trabajo del 100% proporcionaría en el input una señal de 10V. Si en el caso de existe un ciclo de trabajo del 25% la señal seria de 10·0,25=2,5V. Si, por ejemplo, el porcentaje es 0% el valor sería de 0V y por último invirtiendo la polaridad podríamos tener señales de -10V con un ciclo de trabajo del -100%.

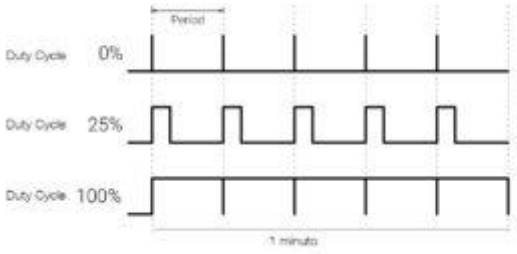



*Figura 12. PWM.*

Para transmitir el movimiento del motor se utiliza mecanismos mecánicos como pueden ser engranajes, poleas, correas de distribución y husillos sin fin entre otros.

### 3.1.3. Selección.

Para construir la máquina se va a escoger de tipo monoespacio con accionamiento electromecánico. Se ha escogido una estructura monoespacio debido a que se busca reducir dimensiones, y el accionamiento electromecánico, ya que controlar cargas más pequeñas y es más fácil mediante este mecanismo, además de facilitar la creación del software.

Cabe destacar que mientras con Arduino se ha controlado la máquina de ensayo construida, mediante la utilización de una tarjeta de Adquisición de NI permitiría gobernar cualquier máquina de ensayo universal que se deseé con el Software desarrollado. Debido a que ambos accionamientos trabajan con acondicionadores que se pueden controlar mediante una señal analógica de entrada de ±10V al servomotor amplifier (electromecánica) o al controlador de la servoválvula (hidráulica) permitirá generar una salida de potencia en forma de PWM que moverá el motor y el movimiento del torque motor respectivamente. Este será máximo en con un voltaje de 10V, con un voltaje 0V estará parado y con una señal de -10V el máximo en la otra dirección.

## 3.2 Estructura mecánica.

La máquina universal de ensayos a crear se deberá parecer a la mostrada en la figura 13. Las partes de esta máquina de arriba abajo son:

1. En forma de S, se observa una célula de carga, su funcionamiento estará explicado posteriormente.

2. Las mordazas se representan como número 4 y 5, necesarias para poder apoyar el material a ensayar. En cambio, la mordaza 5, está colocada en una placa móvil, este movimiento permitirá aplicar la fuerza en ambos sentidos, compresión y tracción. Para un uso como podría ser flexión a tres puntos, se deberían cambiar las mordazas a puntos de apoyo.
3. El mecanismo de arrastre se va a mover, pero consiste en el mismo fundamento. El número mostrado en 1 es un tornillo de potencia, el número 2 consiste en una corona sin fin y el motor es el número 3. La realización de la máquina consistirá de dos husillos sin fin paralelos a las columnas, que muevan la placa móvil. Ambos husillos serán accionados mediante un mecanismo de poleas accionados a su vez por un motor.
4. Además, para poder colocar la electrónica se realizará una especie de mesa donde descansará también la máquina definida anteriormente.

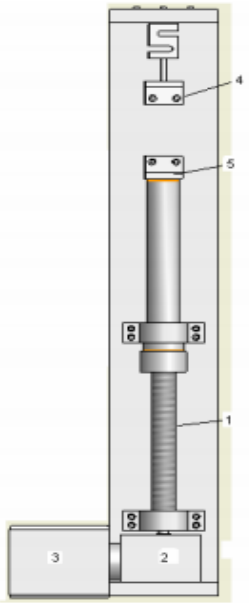



*Figura 13. Máquina universal de ensayos para polímeros.*

## 3.3 Electrónica.

Para la realización de la máquina de ensayos universal son necesarios diferentes tipos de componentes electrónicos, sensores y actuadores. Los elementos más importantes están explicados en los siguientes puntos.

### 3.3.1 Célula de carga y galgas extensométricas.

Una célula de carga y las galgas extensométricas son elementos mecánicos que al ser sometidos a un esfuerzo sufren una deformación del material, que provoca una variación de su resistencia interna. Este tipo de sensor está formado por un puente de Wheatstone, figura 14, un puente de simple montaje donde hay cuatro elementos resistivos conectados en forma de puente, una señal de excitación en una de las diagonales y un lector de corriente o tensión. Este lector mide la diferencia entre las dos salidas. [3] [12]

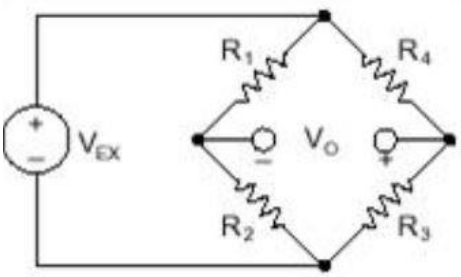



*Figura 14. Puente de Wheatstone.*

Para la salida de la célula de carga, se utiliza un amplificador operacional en modo diferencial. Esto nos permite tener una referencia de tensión de precisión, una tolerancia de la temperatura, aumentar su linealidad, además de aliviar el ruido.

Los cambios de temperatura crean más cambios que de fuerza, por lo que se necesita o bien aislar la célula de carga correctamente, o colocar una galga extensiométrica libre. Esta galga, al no estar afectada por la fuerza, únicamente variará según la temperatura, gracias a ésta se puede conocer el valor de fuerza neto. [11]

### 3.3.2 Motor paso a paso.

Un motor paso a paso es un dispositivo electromecánico que convierte pulsos eléctricos en movimientos mecánicos discretos. El eje de un motor paso a paso gira en incrementos discretos cuando impulsos de mando eléctrico se aplican a él en la secuencia correcta. [11]

La secuencia de pulsos está relacionada directamente con la dirección de rotación, la velocidad con la frecuencia de estos pulsos y la duración de la rotación con el número de pulsos.

Se caracteriza por su capacidad de ser controlado con precisión y además no necesita de retroalimentación de posición por lo que puede trabajar tanto en lazo cerrado como abierto.

Los pulsos eléctricos corresponden al estado encendido o apagado en el cual se encuentran las bobinas del motor, este consta de cuatro. Por lo que encender la primera bobina será un paso, encender la primera y la segunda se realizará otro paso. En la figura 16 se puede observar claramente cómo funciona. [4][5]

Para simplificar el uso de un motor paso a paso se utiliza un chip amplificador, en nuestro caso un el componente VMA409 figura 15, contiene el chip 298. [6]

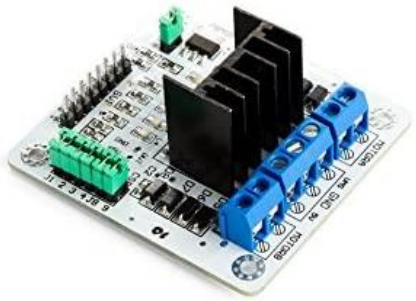


*Figura 15. Chip controlador VMA409*

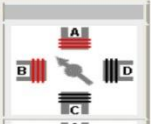
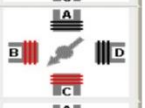
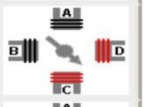
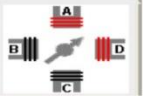
| PASO | Bobina A | Bobina B | Bobina C | Bobina D | |
|---|---|---|---|---|---|
| 1 | ON | ON | OFF | OFF |  |
| 2 | OFF | ON | ON | OFF |  |
| 3 | OFF | OFF | ON | ON |  |
| 4 | ON | OFF | OFF | ON |  |

*Figura 16. Pulsos de un motor paso a paso.*

### 3.3.3 Encoder.

Los Encoders, figura 17, son sensores que generan señales digitales en respuesta al movimiento. Estos son utilizados para medir movimientos, velocidad y posición.

*Figura 17. Encoder montado a la izquierda, y desmontado a la derecha.*

Existen de dos tipos incrementales y absolutos. En este proyecto se utiliza un incremental, que contienen un número específico de pulsos espaciados equitativamente por cada revolución. Para que el encoder pueda diferenciar el sentido de giro, necesita comparar dos canales desfasados 90º.

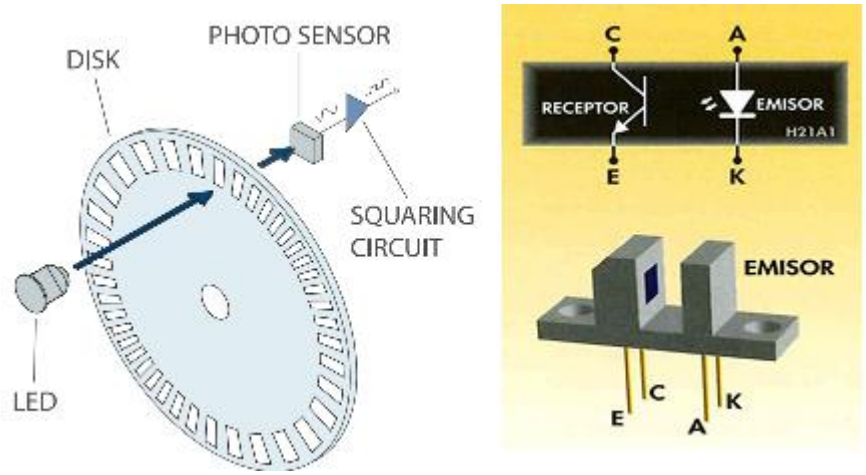



*Figura 18. Encoder.*

En la figura 18 podemos observar el funcionamiento del encoder incremental óptico. La colocación del disco entre el fotosensor y el led provoca una señal sinusoidal gracias a los huecos del disco. En cada paso el sensor deja de recibir luz y vuelve a recibir. El encoder mecánico funciona parecido, en cambio funciona con interruptor, que detecta cada cambio de posición. [11]

### 3.3.4 Placa de control Arduino.

En este proyecto, el dispositivo de control es una placa Arduino. Esta placa electrónica de hardware libre incorpora un microcontrolador reprogramable con unos pines hembra, un botón de reset, una conexión para una fuente de alimentación externa y una conexión para el ordenador.

## 3.4 Procesamiento de señales.

Para realizar una correcta interpretación de los datos obtenidos por los sensores se debe aplicar un regulador de señal. Como regulador se va a utilizar un PID, se necesita por varios motivos:

- Para eliminar el ruido: toda señal no deseada detectada mezclada con la señal útil.
- Mejoran el tiempo de respuesta.
- Mejora la precisión de la respuesta.
- Eliminan errores previos de los dispositivos electrónicos instalados.
- Ajusta y predice la señal.

Este PID se necesita sobre todo en aquellos ensayos donde se usa el control por Carga. Por ejemplo, la máquina se encuentra sin realizar fuerza y quiere llegar a 100N. La gráfica Newtons/s se vería tal y como se muestra en la figura 19.

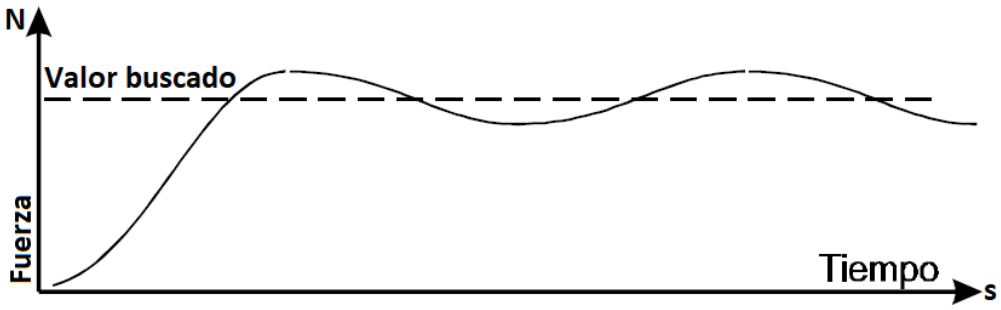



*Figura 19. Fuerza sin PID.*

Debido a la inercia de la máquina, el valor de la célula de carga estará continuamente fluctuando entorno al valor buscado. La corrección hará que el resultado se muestre como en la figura 20. [18]

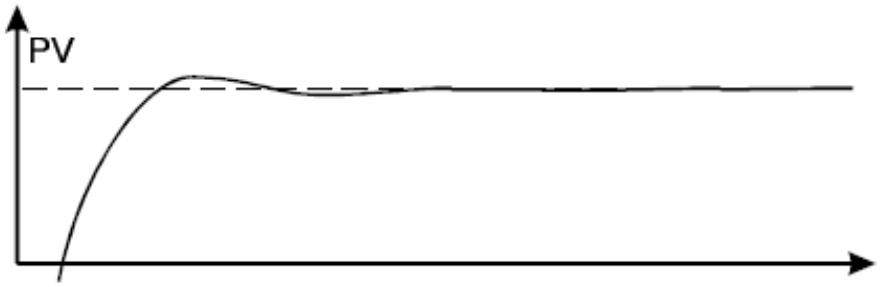



*Figura 20. Señal corregida.*

## 3.5 Arduino.

Arduino es una plataforma de creación de código abierto, basada en software y hardware libre y flexible.

Su éxito reside en general por su bajo coste, ya que por un precio reducido se puede obtener una herramienta que te permite controlar cualquier tipo de electrónica, además de la posibilidad de poder comerciar diferentes proyectos, como podría ser este.

Arduino cuenta con una extensa comunidad, lo que hace que haya innumerables tutoriales, guías y ejemplos en la web para poder realizar proyectos propios desde el autoaprendizaje.

### 3.5.1 Librería Arduino.

Las librerías son códigos ya hechos por terceros que se utilizan en nuestro programa. Esto facilita la programación, hace que el código sea más corto por lo que es más fácil de realizar y entender.

Arduino dispone de un gran número de librerías, todas ellas son gratis y se pueden acceder y modificar. Los proveedores de chips electrónicos pueden subir su "Firmware", que consiste en su librería Arduino, para que cualquiera la pueda descargar y utilizar. Por ejemplo, en nuestro caso utilizamos librerías como la del amplificador de la célula de carga "HX711.h", del encoder o del motor paso a paso <Stepper.h>. [9]

## 3.6 LabView.

### 3.6.1 Introducción al Labview.

Laboratory Virtual Instrument Engineering Workbench (LabView) es un lenguaje de programación visual creado por National Instruments. Este tipo de programación utiliza bloques gráficos con diferentes funciones que, al conectarlos entre ellos, generan un enlace de ejecución de izquierda a derecha y de arriba a abajo. Además, soporta diferentes utilidades como bucles, funciones de lectura y conversión de magnitudes.

LabView se usa como adquisición de datos, instrumentación y control industrial. Además, destaca por su larga lista de drivers compatibles en numerosos dispositivos, con diferentes herramientas, como por ejemplo LINX utilizado en este proyecto para la comunicación con Arduino.

La interfaz de LabView se divide principalmente en dos ventanas:

1. **Diagrama de bloques.** Lugar donde reside el código. Consta de una paleta de bloques con diversas funciones, links que permiten la conexión entre ellos y diferentes utilidades como son los bucles.
2. **Front panel.** Interfaz del usuario, contiene diferentes controles e indicadores que permiten al usuario interaccionar con el código. Por lo general, en los controles se generan inputs y en los indicadores se reciben los outputs, en algunos casos los indicadores pueden servir para comprobar el funcionamiento del código.

# 4. Construcción Máquina de ensayo universal.

El resultado del ensamblaje se puede observar en la figura 21.

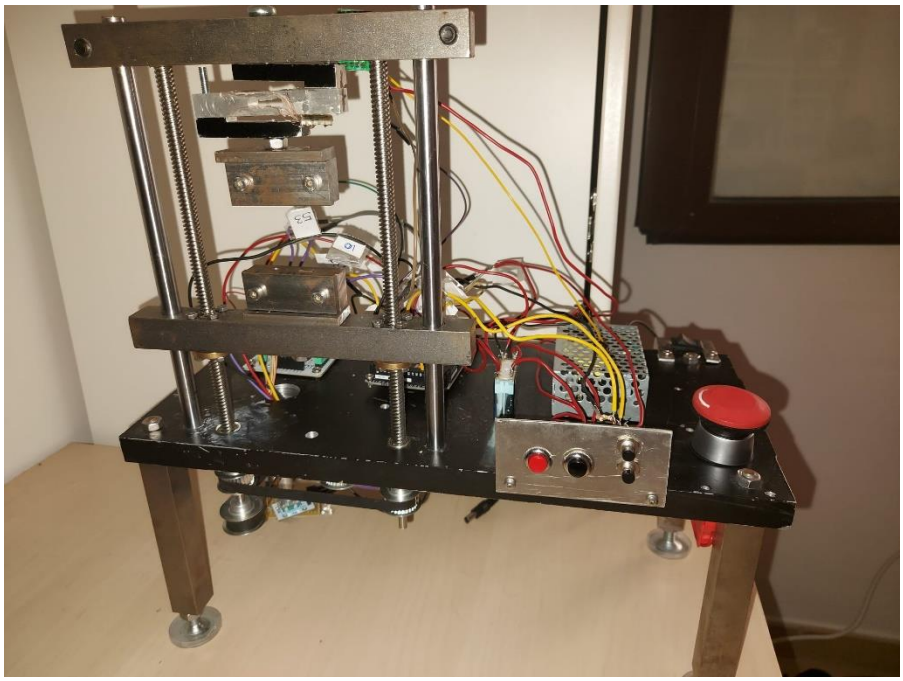


*Figura 21. Máquina universal de ensayos.*

## 4.1 Funcionamiento de la máquina de ensayo universal.

Para la creación de la interfaz se necesita aprender ambos lenguajes de programación. Esta interfaz constará de una comunicación entre LabView y Arduino, dónde LabView funcionará de cerebro enviando y procesando la información que Arduino genera a partir de los componentes electrónicos.

El funcionamiento de la máquina será el propuesto por el siguiente algoritmo:

Primero de todo, el usuario debe elegir los parámetros del experimento y las variables que pueda tener en el Front Panel. Una vez se han colocado todas, se deberá iniciar el experimento mediante el botón Start. Nuestra máquina funcionará mediante una comunicación de 4 señales, dos señales de inputs que serán el sentido de giro y la velocidad del motor, generadas por Labview, y dos señales de outputs la fuerza por la célula de carga y el desplazamiento del encoder, generadas por Arduino. Cada uno de los Software se dedicará a procesar la entrada de las dos señales para obtener los outputs correspondientes. Labview se encargará de acondicionar los valores para poder graficarlos y guardarlos en un archivo Excel.

Cabe destacar que este Software no está diseñado únicamente para la máquina construida. Este podría utilizarse en otras máquinas servo hidráulicas realizando pequeños ajustes. El cambio más significativo es cambiar el SubVi Data, subprograma explicado posteriormente que permite la comunicación entre LabView y Arduino. Este SubVi Data contiene un tipo de comunicación específica con Arduino, para controlar una máquina servo hidráulica únicamente se debe cambiar esta comunicación, ya que no se trabajará con Arduino sino, por ejemplo, con una Tarjeta de Adquisición de LabView como utilizan algunas máquinas de la facultad. A su vez será necesario implementar dodo el sistema de entradas y salidas digitales para conectar alarmas, límites y botones necesarios para su correcto funcionamiento. Por lo tanto, la creación de este programa permite reciclar máquinas de la universidad que el programario ya estaba anticuado.

## 4.2 Creación de la Interfaz.

Para crear una interfaz sencilla y óptima, debemos colocar sólo aquello que sea exclusivamente para la realización del experimento. Además, el usuario debe ser capaz de controlar todos los aspectos del experimento. También, debe existir compartimientos diferentes para cada tipo de variable, por ejemplo: las variables que definan la geometría de la probeta estén separadas de las que definen el tipo de ensayo, así poder crear un circuito que se deberá rellenar. Por lo que la idea de la interfaz inicial será la mostrada en la Figura 22.

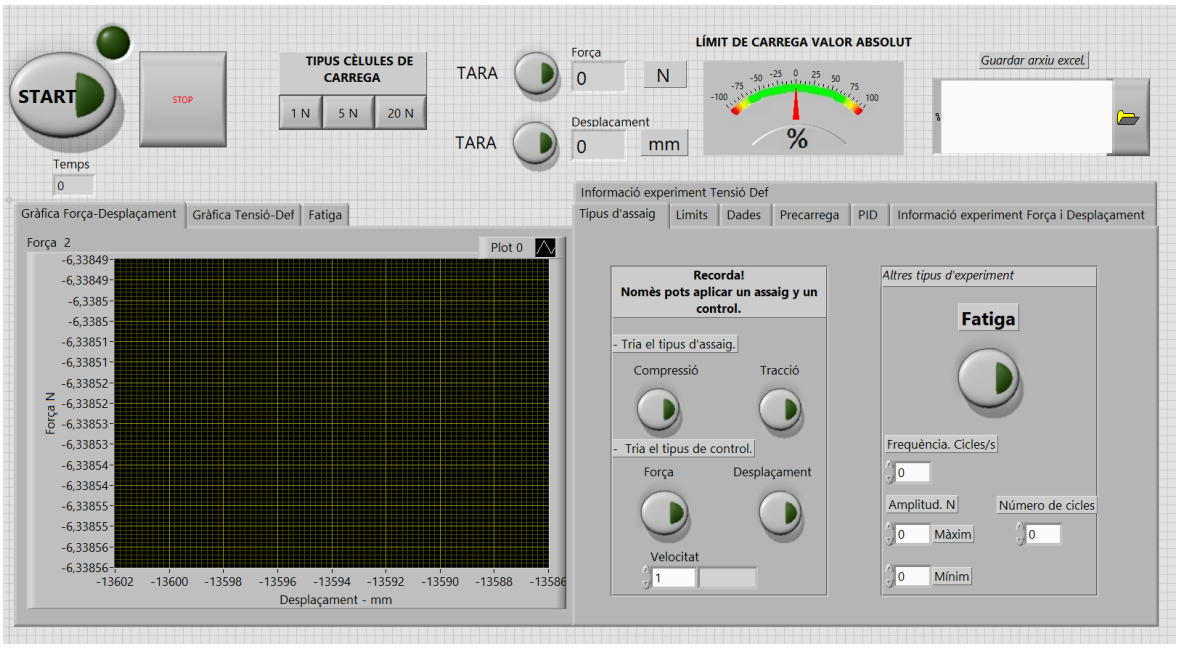



*Figura 22. Interfaz inicial Labview.*

La interfaz la dividiremos en 3 partes.

### 4.2.1 Parte superior interfaz.

En la parte superior se ha colocado aquella información que es relevante observar durante todo el experimento, y aquellos controladores y pulsadores más importantes.

En primer lugar, se encuentra Start y un led. El botón Start servirá para dar comienzo al experimento, y el led indicará el estado de la máquina. Justo debajo, nos encontramos un indicador de tiempo, que indica los segundos que han pasado desde que el botón Start se ha pulsado. A la derecha, tenemos el STOP, el botón que para el programa, haya sido clicado Start o no.

A la derecha están los pulsadores que controlan la célula de carga que se está utilizado. Es completamente necesario la utilización correcta de éstos, debido a que si por confusión se clica una célula de carga más grande de la que realmente se utiliza, la máquina no podrá evitar que se pueda romper.

Más a la derecha están los indicadores de fuerza y desplazamiento. Estos funcionan desde el primer momento, para indicar si las mordazas están realizando ya fuerza o no. Incorporan un valor de tara que permite poner a cero cualquiera de las dos magnitudes todo el experimento.

A continuación, se encuentra un indicador de límite absoluto de carga. Éste comunica al usuario que tanto por cierto de la capacidad de la célula de carga se está utilizando.

Por último, se encuentra un espacio donde se permite al usuario escoger un archivo Excel dónde guardar el experimento. Es completamente necesario, ya que, si no, no se podrán sacar conclusiones del ensayo realizado.

### 4.2.2 Parte bloque izquierdo.

Esta parte nos encontramos un bloque con diferentes pestañas que no se pueden modificar por el usuario. Estas pestañas contienen diferentes gráficas que ayudarán a observar “in-situ” la realización del ensayo. Las gráficas son:

- Fuerza vs. Desplazamiento.
- Tensión vs. Deformación.
- Fatiga: Amplitud vs. Tiempo.

### 4.2.3 Parte Bloque derecho.

El bloque derecho contiene diferentes pestañas, que si se pueden modificar. Estas contienen controladores numéricos y booleanos necesarios para realizar el ensayo.

#### 4.2.3.1 Tipo de ensayo.

En esta pestaña se permitirá al usuario elegir qué tipo de ensayo quiere, si a tracción o compresión y si fuerza o desplazamiento. Además, se deberá indicar el parámetro que controla el tipo de control, N en control por fuerza o mm/min en control por desplazamiento.

Justo a la izquierda, se encuentra la selección de un ensayo de fatiga. Para realizar el ensayo a fatiga se deberá clicar el botón de Fatiga y rellenar los valores correspondientes. Además, se puede escoger un tipo de control, ya que dependiendo de que experimentos de fatiga interesa controlar desplazamiento, y en otros experimentos fuerza.

#### 4.2.3.2 Límites.

En este apartado el usuario deberá colocar los límites oportunos de carga y desplazamiento. Evitando así que las mordazas colisionen entre ellas o con otras partes de la máquina, y que no se sobrepasen fuerzas no deseadas. En cada numérico se encuentra un led al lado. Este Led indicará si el límite no se ha sobrepasado (verde) o de lo contrario si (rojo).

#### 4.2.3.4 Datos.

En este apartado el usuario puede colocar las dimensiones de la probeta.

#### 4.2.3.5 Precarga.

Esta pestaña contiene de un pulsador, que al presionarlo se ejecuta una precarga. La precarga será del valor que se proporcione justo a la derecha del pulsador. Se sabe que la precarga ha finalizado, debido a que el led correspondiente a precarga aplicada se encenderá.

#### 4.2.3.6 PID.

En este apartado se encuentran los valores que regula un PID. Juntamente con ellos, se colocan valores proporcional, integrativo y derivativo. Estos valores podrán ser modificados para poder ajustar la máquina a los cambios en las condiciones de ensayo que se puedan producir.

#### 4.2.3.7 Información experimento.

Éste consta de dos pestañas que nos permiten observar cómo cambian los valores en el tiempo, mediante gráficas e indicadores.

Además, permite ver la velocidad a la que se está realizando el ensayo.

## 4.3 Construcción de la comunicación Arduino-Labview.

Como se ha dicho anteriormente, la funciones de Arduino son:

1. Recibir órdenes de LabView.
2. Mover el motor paso a paso.
3. Interpretar los valores de la célula de carga y del encoder.

### 4.3.1 Recibir órdenes de LabView.

Arduino consta de una ventana de comunicación donde el usuario puede leer y escribir datos, valores, información, etc. Esta ventana será el canal de comunicación de ambos programas. Para que Labview pueda leer y escribir como si fuera un usuario externo es necesario instalar el Add-on LINX (en inglés Add-on se conoce como una extensión o complemento que permite ampliar las funciones de un programario). Esta comunicación se llama comunicación Serial. Arduino utilizando diferentes funciones como son Serial.Read() o Serial.Write("X") puede escribir y leer en este canal de comunicación.

Una vez se sabe por dónde se va a comunicar se debe saber el qué.

### 4.3.2 Motor paso a paso.

Para mover el motor paso a paso nos debemos fijar en los ejemplos que propone Arduino y el fabricante. En nuestro caso para poner en funcionamiento el motor se han utilizado dos funciones de la librería <Stepper.h> en ella necesitamos dos funciones con sendas variables myStepper.setSpeed(b) y myStepper.step(a).

En el caso de la primera función, permite determinar la velocidad a la cual se moverá el motor en pasos segundo. En cambio, la segunda función dará información de la dirección de giro siendo 1 en el sentido de las agujas del reloj, -1 en el sentido contrario y 0 parado. Cualquier valor diferente a 1 o -1, mediante Labview, hará que el motor no se mueva interpretándose como 0.

### 4.3.3 Interpretar valores de la célula de carga y del encoder.

Tanto la célula de carga como el encoder tienen una librería especial dónde se encuentran sus funciones principales. La función que permite leer el valor de la célula de carga es scale.read(), para el encoder se necesita utilizar una interrupción: en el caso que se cumpla una condición este para el programa ejecuta una instrucción y retorna al punto del programa en que se encontraba antes de ésta, en este caso encoderPoscount++, que suma una unidad cada vez que el encoder realiza un paso. En el

Arduino se encuentran los pines de interrupción, que realizan una función esencial para contar siempre todo los pasos dados por el encoder.

### 4.3.4 Conclusión.

Una vez obtenido el qué y por dónde, ahora se debe buscar una manera de como se va a realizar. Un programa que envíe dos variables separadas y que el Arduino deba esperar a recibir una para recibir la otra y lo mismo en sentido contrario no es viable. Por lo que se propone enviar dos valores en forma de uno.

LabView contiene funciones que permiten identificar dos valores separados por un patrón característico, y mediante Arduino también se puede realizar gracias a una lectura en bucle de la información recibida. Por lo que se decide enviar los valores de tal manera: X; Y. Así Arduino podrá separar los valores X de los Y y exactamente igual LabView.

## 4.4 Realización del programa de comunicación.

Una vez conocido que funciones se necesitan y qué necesitamos para enviar y recibir los valores deseados, se procede a realizar el programa. El código lo podemos observar en la figura 23.

```
// MOTOR
#include <Stepper.h>
#define STEPS 100
int vel =10;
const int stepsPerRevolution;
Stepper myStepper(STEPS, 53, 43, 52, 51); // Pins motor. 10, 9, 8, 11
int a; // Sentido del giro.
int b = 1 ; // Velocidad
int c;  // Variable libre.

long reading;
// Celula de carga
#include "HX711.h"
const int LOADCELL_DOUT_PIN = 34; //4
const int LOADCELL_SCK_PIN = 28; //5 Pins Celula de carga
HX711 scale;

// Lectura
String str = "";
const char separator = ';';
const int dataLength = 3;
int data[dataLength];

// Encoder
int pinA = 35;  // 3 Connected to CLK
int pinB = 37;  // 2 Connected to DT
volatile int encoderPosCount = 0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(pinA,INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(pinA),doEncode,CHANGE);
}

void loop() {
  if (Serial.available() > 0)
  {
    str = Serial.readStringUntil('\n');
    for (int i = 0; i < dataLength ; i++)
    {
      int index = str.indexOf(separator);
      data[i] = str.substring(0, index).toInt();
      str = str.substring(index + 1);
    }
    int a = data[0];
    int b = data[1];
    int b = data[1];
    int c = data[2];
    if (b < 1) {
      b = 5;
    }
    digitalWrite(47, HIGH); // 7
    myStepper.setSpeed(b);
    myStepper.step(a);
    if (a > 0) {
      encoderPosCount ++;
    }
    if (a < 0) {
      encoderPosCount --;
    }
    long reading = scale.read();

    Serial.print(reading);
    Serial.print(";");
    Serial.print(encoderPosCount);
    Serial.print(";");
    Serial.println(b);

  }
  while (digitalRead(24) == HIGH) {
    digitalWrite(47, HIGH);
    myStepper.setSpeed(vel);
    myStepper.step(1);
    vel = vel + 0.1;
  }
  while (digitalRead(22) == HIGH) {
    digitalWrite(47, HIGH);
    myStepper.setSpeed(vel);
    myStepper.step(-1);
    vel = vel + 0.1;
  }
}
void doEncode(){
  delay(5);
  if (digitalRead(pinA)==digitalRead(pinB)){
    encoderPosCount++;
  }
  else{
    encoderPosCount--;
  }
  }
```

*Figura 23. Código Arduino.*

El código contiene una primera parte de definiciones, se pueden considerar de la línea 1 a la 31. Esta parte del código define todas las variables, funciones y librerías que se van a utilizar. El cuerpo del código lo vamos a separar en tres partes. La primera parte de la línea 32 a la 44, consiste en la lectura de la comunicación Serial, dónde el programa espera hasta recibir los valores necesarios. Además, consta de un bucle for que realiza la separación de las dos variables separadas por un punto y coma. La segunda parte consiste en el movimiento del motor únicamente corresponden a las líneas 48 a 50, que con las variables obtenidas anteriormente mueven el motor a la velocidad y sentido deseada. Por último, la parte restante es la lectura del encoder y la célula de carga y su posterior envío al serial mediante el Write. Cabe destacar, que el programa fallaba en algunos casos debido a que en algún valor detectaba una velocidad menor a 1, esto causaba que el programa se pausara, ya que es un programa lineal y si no puede realizar una acción se bloquea, En la línea de setup, se define la condición de la interrupción, para evitar que se bloquee, y a partir de esa línea se crea la función de la dicha interrupción

Por lo que respecta a LabView se decidió realizar un subprograma que realice esta comunicación. LabView permite que un programa esté formado de diferentes programas, por lo que se puede ahorrar espacio y "despejar" el código para hacerlo más visual y sencillo de entender. Este subprograma de comunicación, mostrado en la figura 24, lo llamamos SubViData.

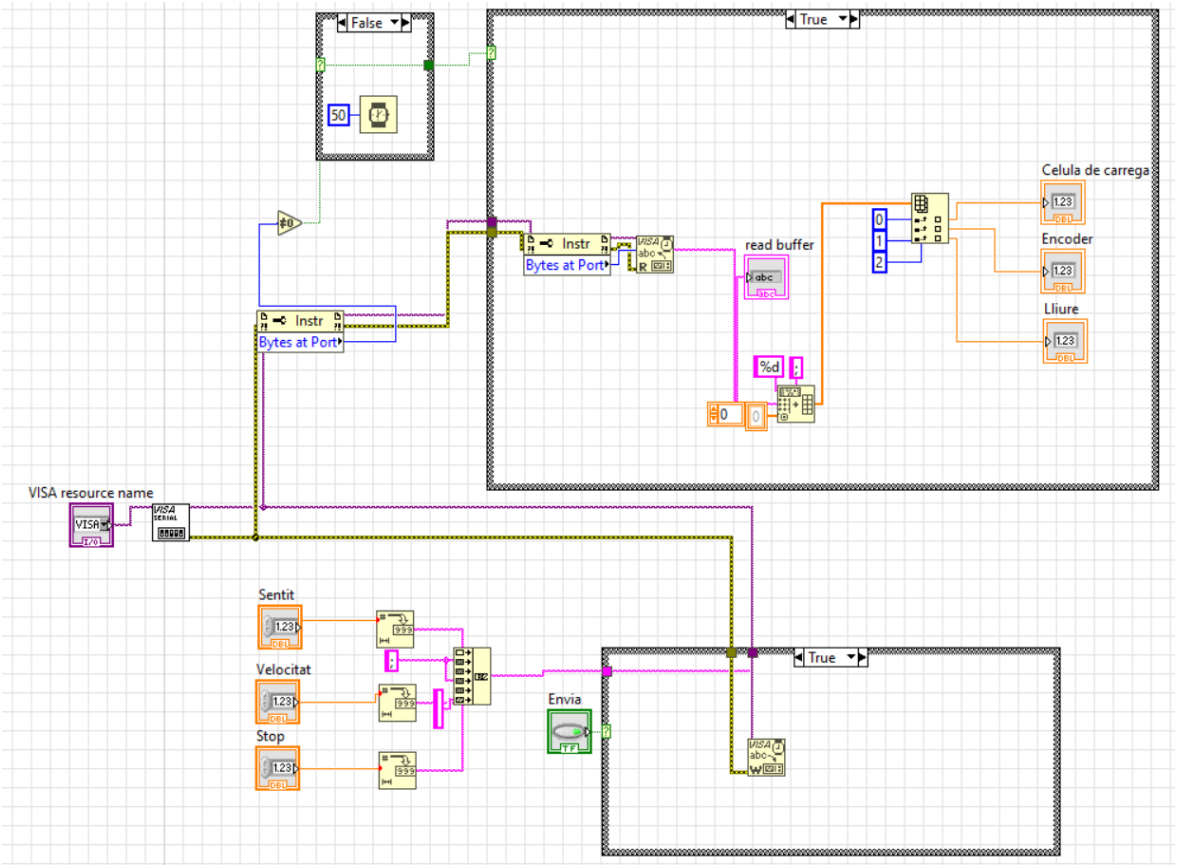



*Figura 24. SubVi Data, comunicació amb Arduino.*

Como dijimos en el apartado de LabView, este software se mueve de izquierda a derecha. Por lo que primero de todo lo que nos encontramos es que abrimos la comunicación serial. Posteriormente realiza simultáneamente dos cosas.

- La primera es si en la ventana de comunicación Arduino ha enviado algún valor, en el caso que sea negativo, Labview omitirá esta parte y no realizará nada.
- La segunda tarea que hará simultáneamente es leer los valores colocados por el usuario de Sentido, Velocidad y Stop (este último se colocó para poder parar desde Labview o por si se necesitará un valor extra en el futuro). Posteriormente, estos valores se unirán utilizando como separador un punto y coma.

En el siguiente instante de tiempo, se comprobará que el pulsador “Envía” está activo, con este botón podemos controlar cuando queremos que Arduino reciba valores o no.

En el caso de que este esté encendido, LabView escribirá los valores de las variables en el Serial.

Este programa se situará dentro de un bucle, por lo que se realizará hasta que el usuario presione el Stop.

En la siguiente interacción hará exactamente lo mismo, con excepción de la posibilidad que Arduino haya escrito los valores obtenidos de la Célula de Carga y del Encoder. En este caso, Labview recibirá ambas variables unidas por el punto y coma. La comunicación se realiza con tipo "String" esto quiere decir que los números para Labview son carácteres, por lo que es obligatorio colocar una función que obligue a Labview que lo lea como números. Hacer este paso, construirá una matriz de números, por lo que posteriormente se debe leer cada posición de esta matriz y se obtendrán los valores.

## 4.5 Software de almacenamiento de datos.

Para almacenar los datos obtenidos de la máquina se deberán seleccionar que datos se desean guardar y dónde. Para eso se ha realizado un programa en LabView sencillo, mostrado en la figura 25. Este programa recibirá dos variables del programa principal y las escribirá en el Excel que se ha seleccionado. Se ha escogido los valores de fuerza, desplazamiento y tiempo para ser guardados. Esto es posible de modificar únicamente conectando los valores de tensión y deformación en el programa principal.

En el programa principal se podrá encontrar la selección del archivo Excel ya creado en la interfaz como se muestra en la figura 26.

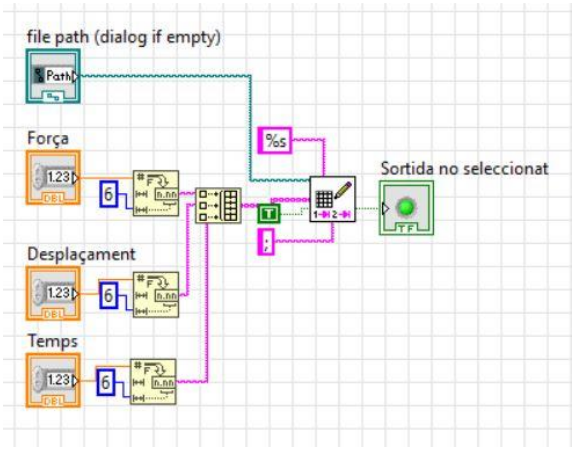



*Figura 25. Entrada datos en Excel vi.*

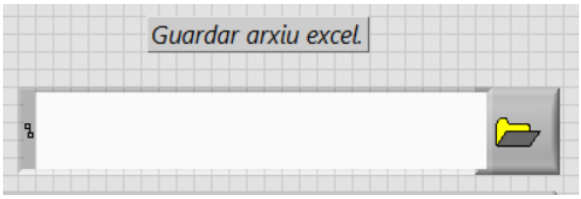



*Figura 26. Interfaz Excel en programa principal.*

## 4.6 Construcción Algoritmo. Programa final.

Para la construcción del programa final se ha realizado un diagrama de flujo, figura 28. Para este proyecto el diagrama de flujo es una representación de los casos del programa. Dónde se establecen los pasos que el programa sigue desde el principio hasta el final, para ello se utilizan diferentes elementos visuales que permiten entender el "flujo" del mismo. Este algoritmo está influenciado por numerosa bibliografía correspondiente a construcciones de máquinas de ensayos universal. [7][8][10]

Su utilización ayuda a: comprender que es cada caso y cuál es su objetivo; facilita posteriores análisis; facilita la posibilidad de mejorar lo y optimizarlo; se identifican pasos críticos.

El diagrama de flujo creado contiene cuatro elementos, figura 27. El primero es el rectángulo, utilizado para describir cada caso y la actividad del mismo. El segundo la elipse, determina el inicio y fin de un caso. El tercero, el romboide que denota un punto de decisión que se debe responder con Si/No. Por último, la flecha esta permite saber el flujo que sigue el programa entre los casos.

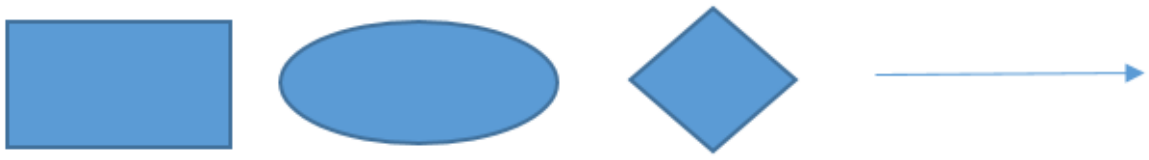


*Figura 27. Elementos del diagrama de flujo.*

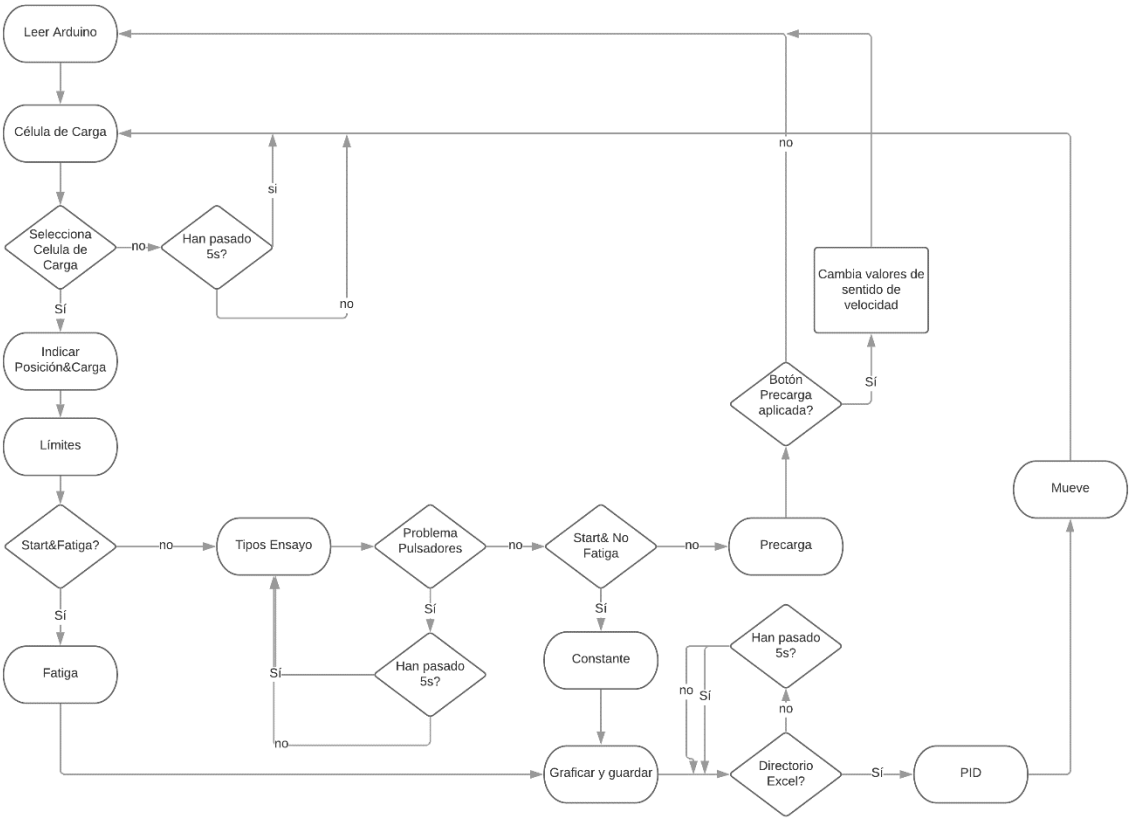



*Figura 28. Diagrama de flujo.*

### 4.6.1 Flujo cero.

En este apartado se realizará la descripción del flujo "cero" del programa principal, este flujo consta de diferentes estados que funcionarán antes de que se empiece el ensayo, es decir, hasta que se presione Start.

Una vez el programa se encienda, este deberá abrir Arduino para permitir la posibilidad del movimiento de las mordazas, algo totalmente necesario para la colocación de la probeta. Por eso el programa empieza a partir de "Leer Arduino". Este caso hace que el programa de Arduino únicamente lea valores, esto solo es posible mediante el envío de valores 0 sentido y 1 de velocidad. El sentido es interpretado por Arduino como la cantidad de pasos a dar, por lo tanto, el valor 0 genera que el motor se pare, en cambio la velocidad 1 es necesaria, ya que el motor no puede interpretar un valor 0 en la velocidad que implicaría que se necesita un tiempo infinito para realizar el paso, por lo que se quedaría bloqueado hasta que no se le quitará la alimentación. En cambio, Arduino si enviará los valores del encoder y de la célula de carga.

Si seguimos con el flujo observamos el cambio de caso a “Célula de Carga”. Este caso realiza dos funciones importantes: la primera, comprueba el estado de carga de las células, es decir, si la carga existente supera el 90% de su capacidad; la segunda función es realizar los factores de conversión convenientes, para poder transformar los valores de la célula de carga en valores de Fuerza.

Después de este caso se encuentra un camino hacia el punto de decisión “¿Se ha seleccionado una célula de carga?”, en el caso de que más de una célula o ninguna esté seleccionada, se dirigirá a otro punto de decisión, dónde se esperan 5 segundos, pasado este tiempo se envía un mensaje de: “Selecciona una célula de carga!”. Tanto como si se pasan los cinco segundos o no el programa se dirige de nuevo hacía “Célula de carga”.

En el caso de que hayan seleccionado correctamente la célula de carga, el siguiente caso será “Indicar Posición y Carga”. En este caso, se realiza la lectura de la Fuerza y desplazamiento. Además, se ha añadido la función de poder realizar una tara, así pudiendo eliminar valores residuales de Fuerza en la colocación de la probeta, un cero falso, el peso de las mordazas, o por ejemplo el valor de la precarga.

Posteriormente sigue “Límites”, un caso donde evalúa si los valores de desplazamiento y de fuerza superan a los límites proporcionados por el usuario, así pudiendo evitar cualquier accidente provocado por sobrepasarlos.

Posteriormente, nos dirigimos a dos puntos de decisión:

a. El punto más crítico “¿Se han superado los límites?” también está conectado con “Célula de Carga”. En el caso afirmativo, el programa se delimitará y únicamente realizará la siguiente secuencia: “Leer Arduino” - “Célula de Carga” - “Indicar Posición y Carga” - “Límites” - “Leer Arduino” …
b. El otro punto es “¿Start y Fatiga?”. Si ambos están presionados, el programa se dirigirá a Fatiga. Si la respuesta es No, seguirá con “Tipos de Ensayo”.

“Tipos de Ensayo” realiza una comprobación de los pulsadores, que no existan peticiones por el usuario contradictorias como podrían ser realizar un ensayo de Tracción y de Compresión a la vez. Por eso existe punto de decisión “Problema Pulsadores”, en el caso de que sí que existan contradicciones, el programa después de que pasen cinco segundos volverá al caso de “Tipos de Ensayos” enviando el mensaje “Únicamente puedes tener un tipo de control y ensayo clicado, comprueba los pulsadores de nuevo. Si quieres hacer Fatiga, pulsa únicamente Fatiga.”, en el caso de que todo este correctamente se avanza al punto de decisión “Start y No Fatiga”. En el bucle 0 ambos no estarán clicados por lo que el programa se dirige por la rama No.

Esto sigue hacia el estado “Precarga”, en el caso de que el botón Precarga no este clicado, el programa volverá al punto cero, volviendo a “Leer Arduino”, si a diferencia Precarga está clicado, la ruta es la

misma hacía “Leer Arduino”, pero hasta que el valor de la célula de carga supere o iguale al valor de precarga, “Leer Arduino” moverá el motor. Este recorrido se repite una y otra vez hasta que se cumpla la condición, posteriormente “Leer Arduino” tendrá la misma función que anteriormente, únicamente leer.

### 4.6.2 Flujo Ensayos de Compresión y Tracción.

Este flujo empieza con el flujo cero, pero esta vez se cumplen las condiciones de “Start y No Fatiga”, entre los casos “Tipos de Ensayo” y “Precarga”, por lo que este último no se ejecutará. Cuando se presiona el botón Start se pasa al estado “Constante”.

Posteriormente, se pasa a “Graficar y Guardar” dónde, como bien dice el nombre, se grafican y se guardan todos los datos. En consecuencia, para guardar la información, se necesita que el usuario rellene la dirección del Excel que se quiere modificar, por eso existe el punto de decisión de “¿Directorio seleccionado?”, este punto conecta a otro punto de decisión de “Han pasado cinco segundos?”, en el caso de que, si hayan pasado, el programa preguntará: “Selecciona el archivo Excel dónde quieres guardar los datos del experimento”. En ambos casos se vuelve al caso “Graficar y Guardar”.

Si se cumple la condición, y el directorio está seleccionado, seguirá el caso “Mueve”. Dónde se envía al motor la velocidad constante y la dirección elegida por el usuario. Una vez se obtienen los valores de célula de carga y del encoder, el programa se dirige hacía “Célula de Carga” de nuevo.

### 4.6.3 Flujo Ensayos de Fatiga.

Para entrar en este flujo, después del caso “Límites” el botón Start y de Fatiga deben estar seleccionados. En consecuencia, se avanzará al caso “Fatiga”. Este caso contiene los valores indicados por el usuario de: carga máxima y mínima, frecuencia y límite de ciclos. Estos valores los transforma en una onda sinusoidal modificando los valores de sentido y de velocidad correspondientes para realizar el ensayo correctamente.

Una vez “Fatiga” se ha realizado, el programa seguirá con el caso “Grafica y guarda”, “Mueve” y “Célula de Carga”. Así repitiéndose hasta que se haga clic en Stop o se supera el número de ciclos.

## 4.7 Código principal.

El código principal se puede observar en la figura 29. Para explicarlo explicaremos caso a caso, de izquierda a derecha.

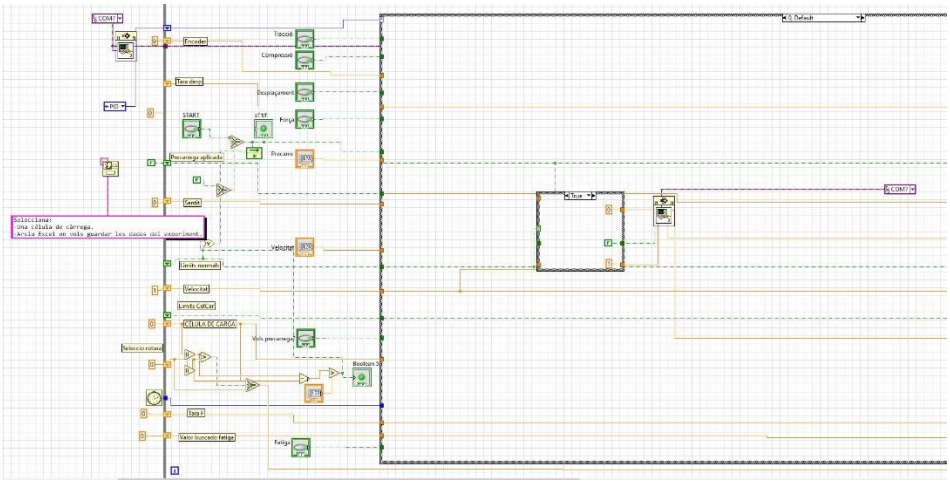


*Figura 29. Código principal.*

### 4.7.1 Parte exterior.

En la parte exterior están aquella parte del código que se ejecutara una vez encendido el programa. Primero tenemos un Enum, una lista de los casos que existen en el interior. Esto permitirá reducir el código en dimensiones y poder ordenarlo correctamente. Más abajo tenemos un SubViData, explicado anteriormente, que se coloca para que el Arduino se abra. Por último, más abajo se encuentra un diálogo. Este diálogo informa que se debe seleccionar una célula de carga y un archivo Excel antes de empezar a realizar cualquier otra cosa.

### 4.7.2 Bucle While.

Un bucle While es un bucle que funcionará hasta que se dé una condición que lo pare. En el caso del software creado, solo se parará en caso de que se presione el botón stop.

Además, el bucle While tiene un recurso muy importante que se ha utilizado llamado “Shift Registrer”. Un Shift registrer, permite guardar valores de ciclo a ciclo, por lo que permite realizar modificaciones, interpretaciones o factores de conversión. En total se han utilizado 9 shift registrer, para valores como encoder o célula de carga, o para otros como pueden ser si se han sobrepasado los límites.

Dentro de este bucle están todos los botones e indicadores, debido a que un botón fuera del bucle While no podrá nunca cambiar de estado. A no ser que se salga del bucle y se vuelva a entrar.

En el bucle While podemos encontrar una operación con el botón Start. Esta operación nos permite que, una vez clicado Start, el programa no pare hasta que se fuerce el paro con el Stop del While o si se han superado los límites. Esto permite como se ha dicho anteriormente en el diagrama de flujo, que una vez superado los límites el programa únicamente lea.

En este bucle While se ha realizado una función que identifique la fractura, mostrado en la figura 30.

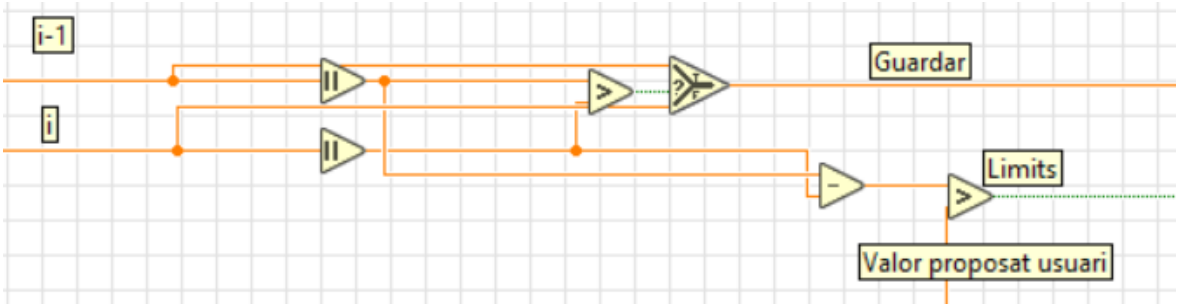



*Figura 30. Rotura.*

Este tiene la función de comparar la lectura de la célula de carga anterior(i-1) con la lectura actual(i). En el caso de que el valor actual sea más pequeño en valor absoluto, el valor de la célula de carga no variará. Además, en el caso de la fractura, se podrá controlar. Esto se hará realizando una resta entre el valor actual y el anterior y comparándolo con un valor propuesto por el usuario de rotura. En caso de que haya superado ese salto, LabView únicamente leera.

### 4.7.3 Case Structure.

El Case Structure permite, tener en una única estructura todo un programa. Este tipo de organización permite mostrar el código que únicamente se desea, y el que no se desea no. En este proyecto se han realizado dos tipos de Case Structure, se consigue un tipo u otro enlazando un valor o bien booleanos o bien Enum. En el caso de booleano se pueden seleccionar los valores verdadero o falso, por lo que sí, una variable es verdadera, se realizará una acción y si en cambio es falsa, se realizará otra. Los Enums en cambio, son unas variables en forma de lista, por lo que el case Structure tendrá tantos casos como la lista contenga.

La lista está compuesta por:

1. Leer Arduino. Figura 31.
2. Célula de Carga. Figura 32 y 33.
3. Indicar PosCar. Figura 34.
4. Límites. Figura 35 y 36.
5. Tipos de Ensayo. Figura 37 y 38.
6. Precarga. Figura 39 y 40.

7. Fatiga. Figura 46 y 47.
8. Constante.
9. Graficar-Guardar. Figura 41,42,43 y 44.
10. PID. Figura 45.
11. Mueve. Igual que figura 31.

#### 4.7.3.1 Leer Arduino.

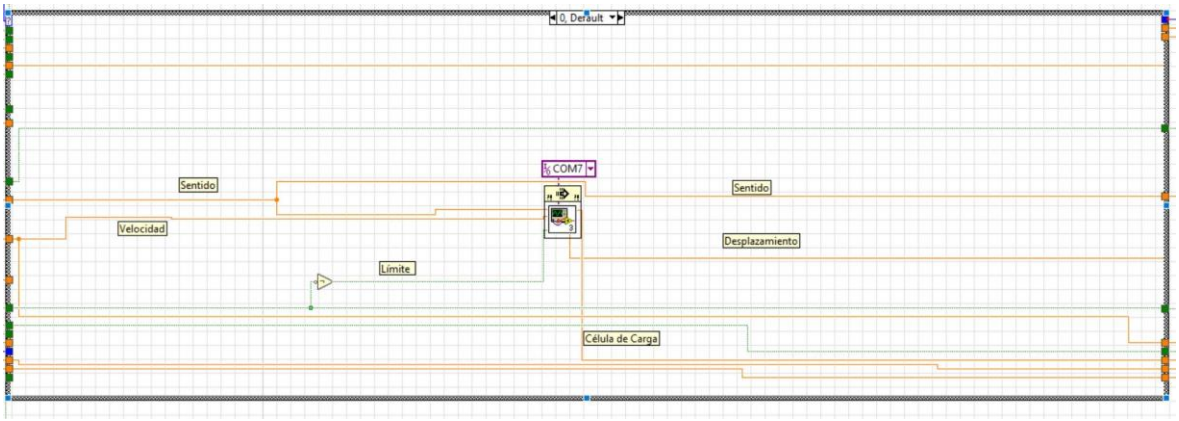



*Figura 31. Leer Arduino.*

Este caso se utiliza o bien para leer, o bien para mover la precarga. Por lo que caben destacar los inputs, Sentido, velocidad que dan información a Arduino de cómo debe mover el motor, y la negación del booleano límite, este indica que, si se han superado los límites, el programa no puede enviar valores, únicamente recibir.

Los outputs son desplazamiento y célula de carga.

Éste programa avanzará a Célula de Carga.

#### 4.7.3.2 Célula de carga.

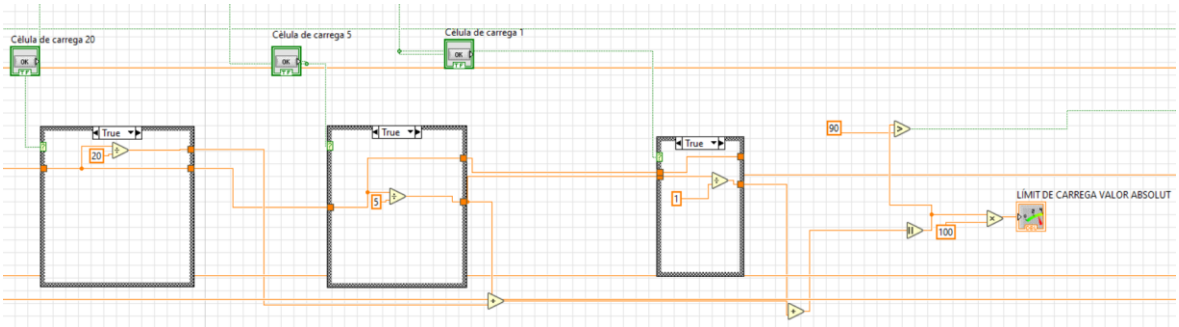



*Figura 32. Primera parte célula de carga.*

Existen dos funciones en Célula de carga, la primera mostrada en la figura X, es comprobar que el límite sea correcto. En cada Case existe un factor de conversión que transforma los valores de la célula de carga en N. También, calcula el tanto por ciento de la carga que se encuentra y se realiza la pregunta de ¿>90%?, en el caso de positivo, este hará saltar el botón Start dejando el programa en modo lectura.

La segunda parte, en la figura X, en cambio comprueba que al menos una célula de carga está seleccionada. En caso afirmativo el programa se dirigirá al siguiente caso, Indicar PosCar, en cambio, si no hay ninguna seleccionada, esperará cinco segundos y enviará un mensaje diciendo que se seleccione una célula de carga, enviando de nuevo el usuario a repetir este caso hasta que se seleccione una célula.

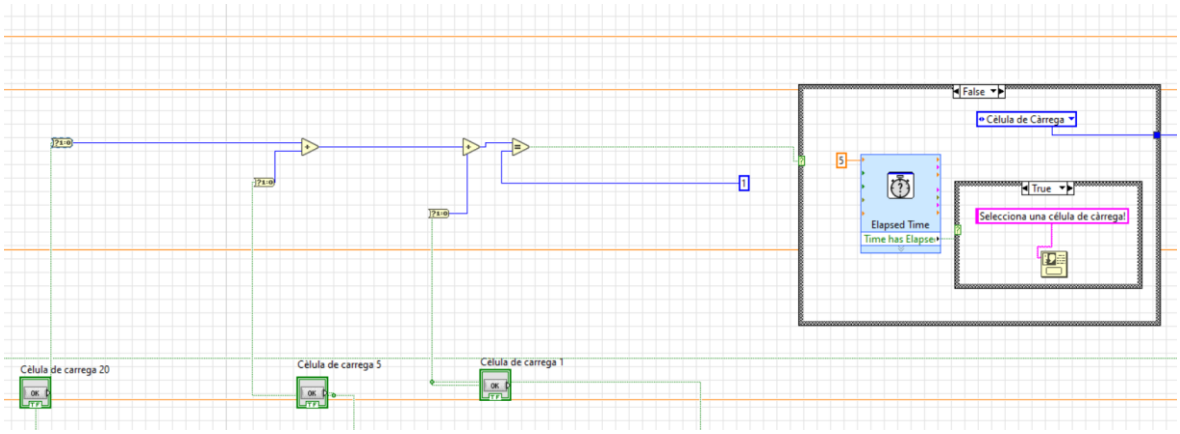



*Figura 33. Segunda parte célula de carga.*

#### 4.7.3.3 Indicar PosCar.

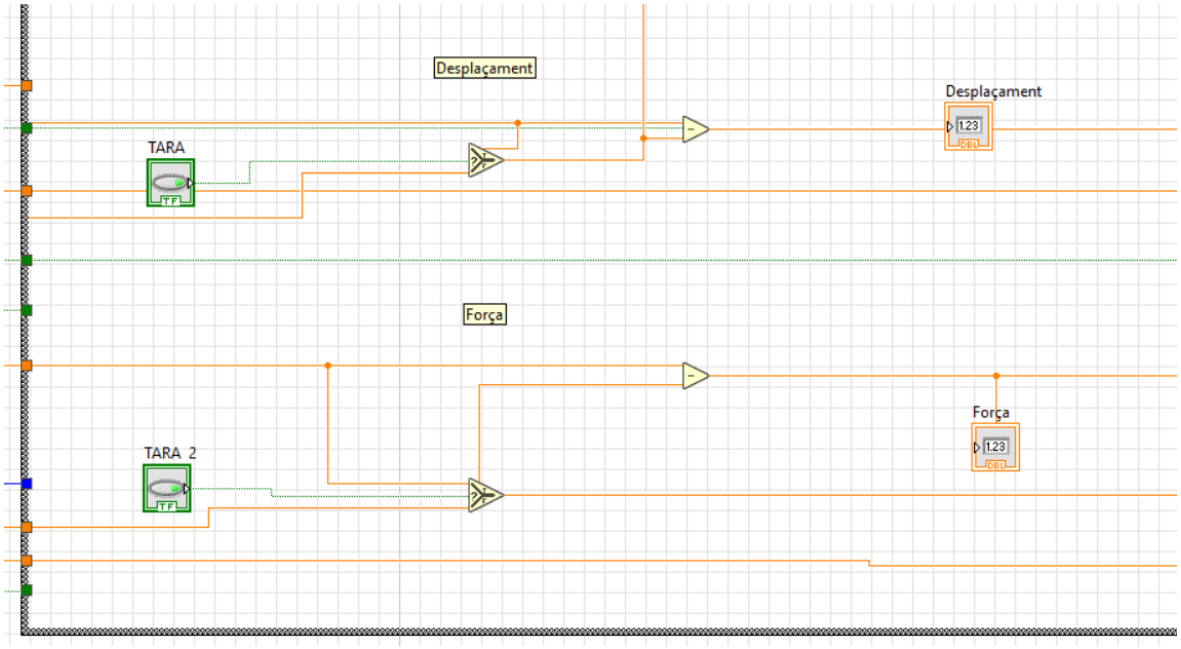



*Figura 34. Indicar poscar.*

Este caso tiene la función de mostrar la fuerza y desplazamiento en cualquier momento del ensayo. Además, cuenta con un comparador que permite la Tara. El valor tarado se coloca en un Shift Registrer y se utiliza posteriormente. El siguiente caso es límites.

#### 4.7.3.4 Límites

Este caso compara los valores obtenidos por el usuario y los límites establecidos. En el caso que se superen, enviará al programa a únicamente modo lectura, figura 31. Además, límites también discrimina el pulsador de Fatiga mediante un Case Structure True/False, por lo que si Fatiga está encendido, el programa se dirigirá a Fatiga. En cambio, si no se ha presionado, irá a Tipos de Ensayo, figura 30.

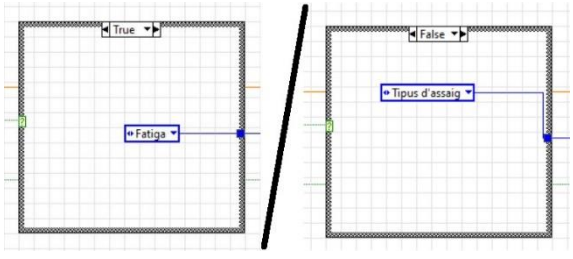



*Figura 35. Fatiga o tipos de ensayo.*

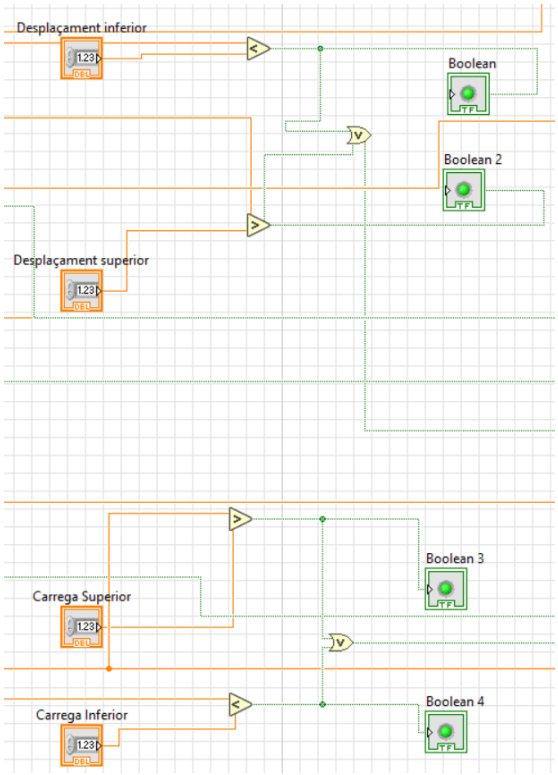



*Figura 36. Límites.*

### 4.7.3.5 Tipo de Ensayo.

Tipos de Ensayo se resume en la siguiente conexión, mostrada en la figura 32.

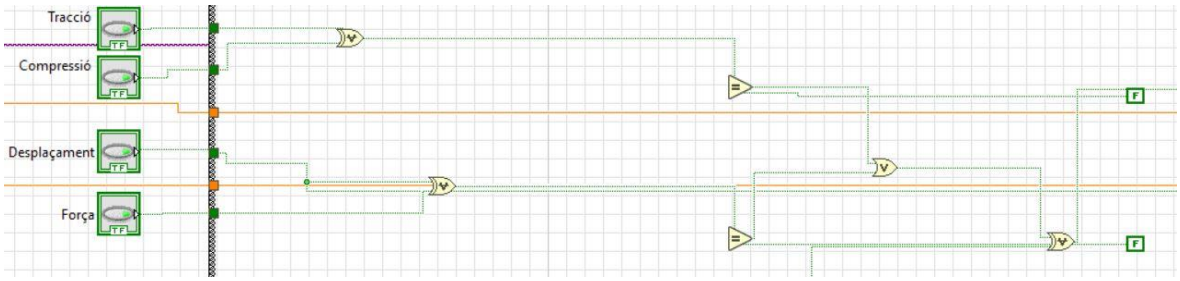



*Figura 37.Conexión tipo de ensayo.*

En resumen, si se observa la tabla de la verdad de esta conexión obtenemos que únicamente es cierto si uno de cada tipo de ensayo y uno de cada tipo de control esta seleccionado. En el caso de que exista más de uno, o ninguno, el programa entrará al Case Structure True de la figura X, donde se emitirá un mensaje anunciando que ha habido un error seleccionando los pulsadores, y volverá otra vez al mismo case hasta que se arregle. En cambio, si esta seleccionado correctamente el programa entrará en el False. Aquí, según el tipo de control que se realice se imprimirán unas unidades de velocidad u otras. También en el caso de que el botón Start esté seleccionado, se avanzará a Constante y en cambio si está apagado, se dirige a Precarga.

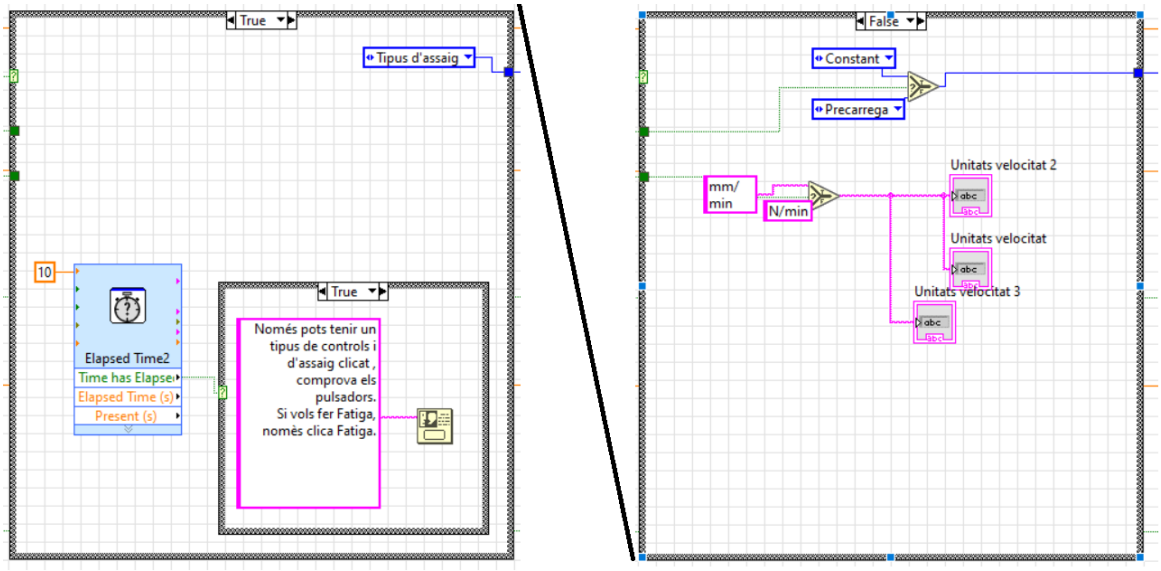



*Figura 38. Tipos de ensayo, case structures*

### 4.7.3.6 Precarga.

En el caso de que aún no esté clicado el botón Start el programa irá a Precarga. En precarga, se observará si el botón precarga está apretado. En el caso afirmativo, se modificarán la velocidad a 5 mm/min, una velocidad estándar y el sentido según el valor de la precarga, figura 40. En el caso de que le precarga sea negativa, el sentido será negativo y al revés en positivo. El programa seguirá el bucle descrito hasta ahora, hasta que en precarga se reciba la orden de que el valor de la célula de carga es mayor o igual que el valor de precarga seleccionado. Posteriormente seguirá el camino hasta Leer Arduino de nuevo.

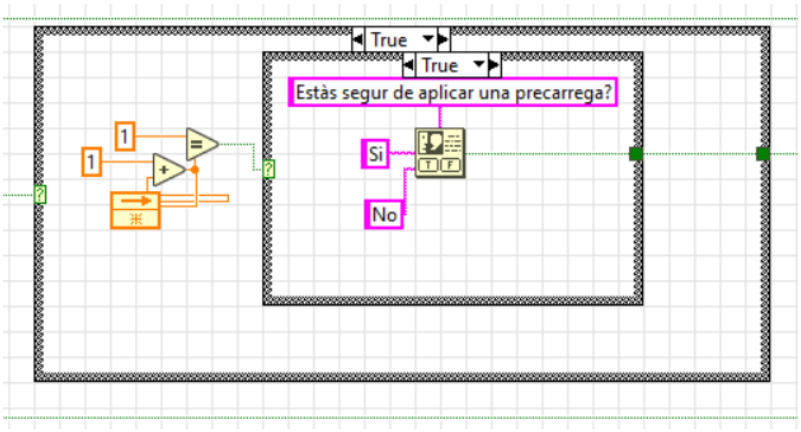



*Figura 39. Pregunta precarga.*

Para asegurar que se quiere realizar precarga se permitirá al usuario cancelar la precarga mediante un dialogo, figura 39. En el caso de que se presione Si, el programa realizará la precarga, en caso negativo el programa no realizará precarga.

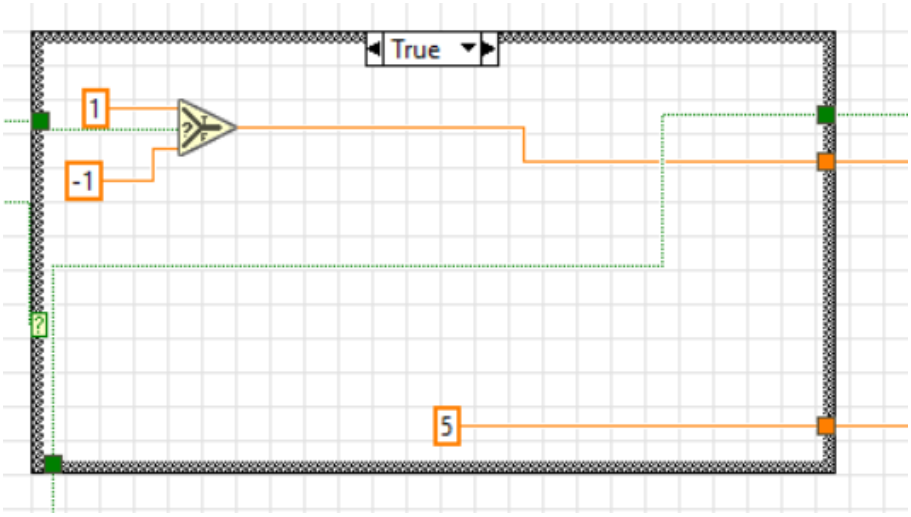



*Figura 40. Valores de velocidad y sentidos de la precarga.*

#### 4.7.3.7 Constante.

En el caso de que el botón Start se haya clicado, el programa avanzará hacia constante. Cuando … el programa se dirigirá a Guardar y Graficar. Este caso se deja vacío para poder dar la posibilidad de añadir funciones nuevas.

#### 4.7.3.8 Guardar y Graficar.

Este caso consta de múltiples gráficas que permiten poder leer el comportamiento "in-situ" de la fuerza y el desplazamiento respecto otras magnitudes, vista completa en la figura 41. El código de las gráficas se puede observar claramente en la figura 42.

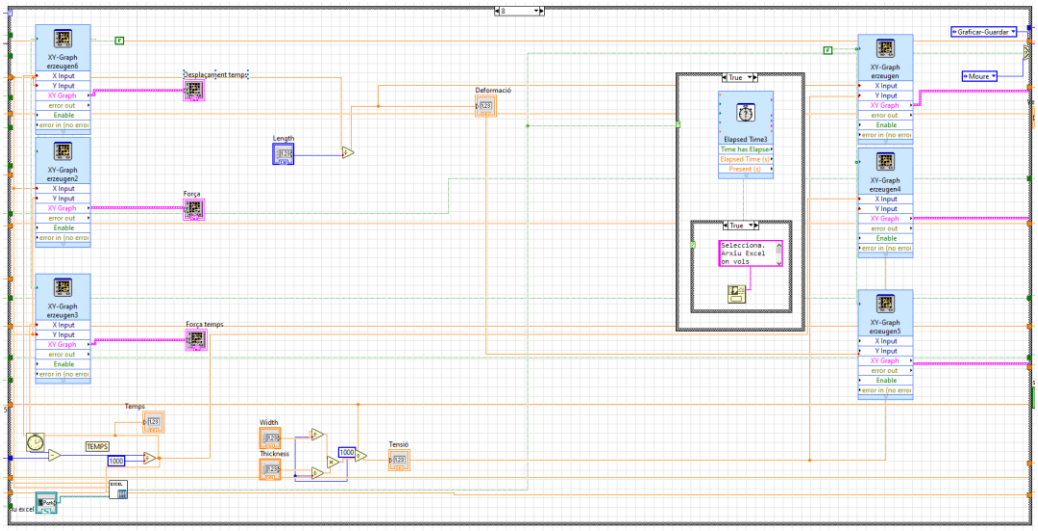


*Figura 41. Guardar Graficar.*

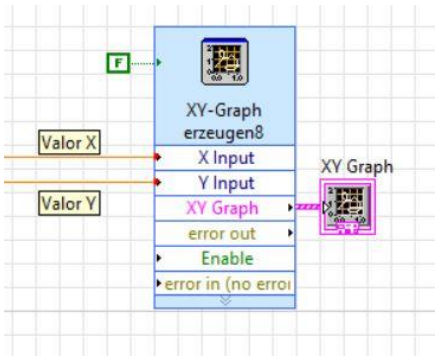



*Figura 42. Ejemplo programación de una gráfica.*

Además, este contiene el subprograma de guardar en Excel explicado anteriormente, con el cableado mostrado en la figura 43.

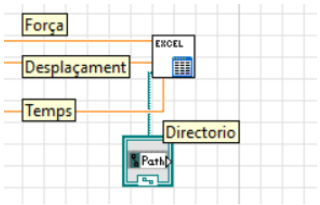



*Figura 43. Cableado subvi Excel.*

Por último, este caso contiene un Case Structure que identifica si se ha seleccionado o no un directorio para guardar el archivo Excel. El caso positivo, figura 44, consiste en notificar después de que hayan

pasado 5s: "Selecciona donde quieres el archivo Excel.". En el caso que sí se haya seleccionado no existe ninguna acción. Posteriormente se avanzará a PID.

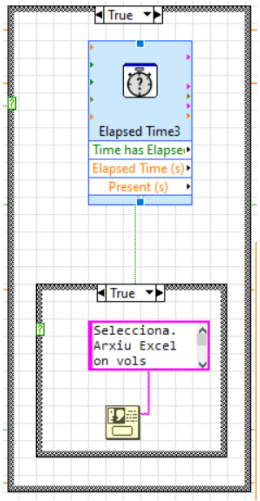



*Figura 44. Selecciona archivo Excel.*

#### 4.7.3.9 PID.

En este caso, se utilizará una corrección de señal explicada anteriormente en Estado del Arte. Para realizarla se llamará a un ejemplo de la librería LabView, este se puede observar en la figura 45. Después se realizará el movimiento con Mueve.

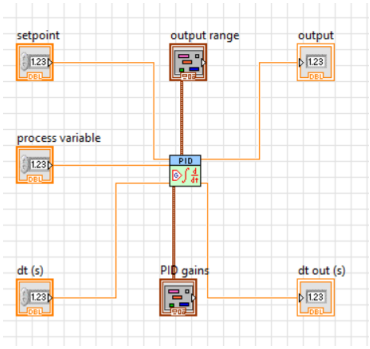



*Figura 45. PID en programa principal.*

#### 4.7.3.10 Mueve.

Es exactamente igual que leer, pero en este caso, gracias a la conexión booleana del Start, sí que envía valores a Arduino. Al acabar el programa, este reiniciará el bucle por Célula de Carga.

#### 4.7.3.11 Fatiga.

En el caso de que en indicar poscar, fatiga estuviera seleccionada, el programa se dirigirá al caso Fatiga. El caso fatiga, figura 46, contiene un generador de señal sinusoidal, que transformará los inputs de valor máximo, valor mínimo y frecuencia en una onda.

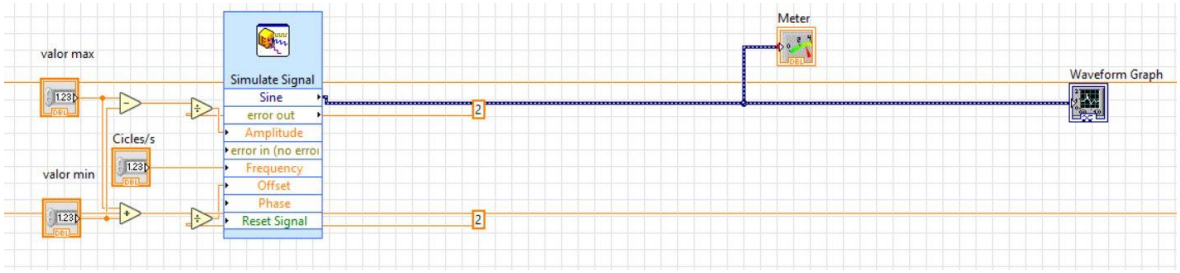



*Figura 46. Caso Fatiga.*

Antes de realizar cualquier acción, el caso fatiga irá contando el tiempo en segundos transcurridos desde que ha empezado el ensayo, multiplicado por los ciclos por segundo seleccionados. Éste valor, conjuntamente con el valor asignado a mº.ciclos por el usuario, harán que en el caso de que sean iguales, el programa pare inmediatamente, figura 47.

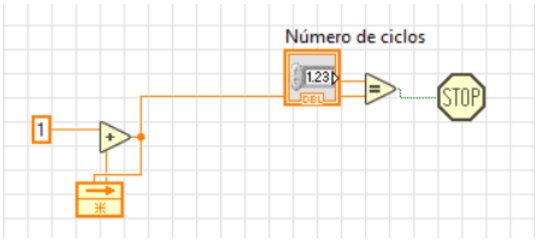



*Figura 47. Stop fatiga.*

Posteriormente, estos valores deben ser enviados y tratados por el PID que es el que se encargará de transformarlos en una consigna de velocidad y dirección para enviar al Arduino.

Este caso al finalizar seguirá con Grafica y Guarda.

# 5. Manual de instrucciones.

## Introducción.

Leer este manual antes de utilizar el programa para realizar un buen uso del mismo. Contiene información sobre el uso seguro y eficiente del mismo. Guarde este manual para que cualquier usuario nuevo o ya experimentado pueda realizar consultas posteriores.

## Peligro.

- Antes de realizar cualquier ensayo mediante este programa asegúrese que está trabajando con las condiciones correctas.
- No permita utilizar la máquina a personas con falta de conocimiento de su uso o falta de experiencia sin supervisión. Las fuerzas empleadas por la misma pueden ser muy elevadas y peligrosas.
- Una vez comenzado el ensayo, no tocar la máquina. Si existe alguna falla, utilice el parado de emergencia de la máquina o asimismo el del programa.

## Advertencias de uso.

- Asegúrese que la temperatura y la humedad son las indicadas.
- Colocar la máquina en un suelo irregular puede variar las lecturas de carga, utilice siempre la máquina en el mismo lugar.
- No cargue la célula carga con un peso superior al que pueda aguantar, si no está seguro consulte a un experto.
- En caso de que la máquina no funcione, al final de este manual encontrará un Troubleshooting dónde encontrará los errores más comunes, en todo caso comuníquese con el servicio técnico para cualquier anomalía.
- Mantenga la máquina limpia y el alrededor ordenado.

## Inicio.

Cuando ejecute el programa verá el mensaje mostrado en la Figura 48, dónde deberá clicar “OK”.

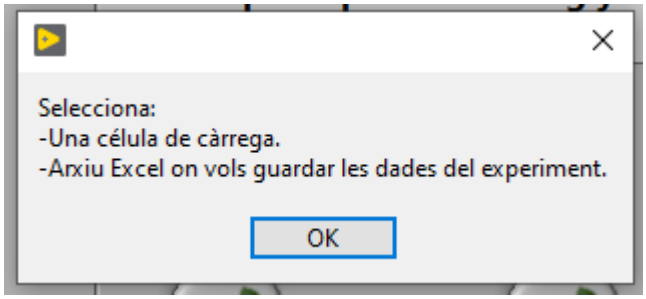



*Figura 48. Mensaje de bienvenida.*

Para continuar, deberá seleccionar los valores de célula de carga y el archivo Excel donde se desea guardar. La célula de carga se debe escoger entre las tres opciones propuestas, en el caso de no saber cuál este colocada, no dude en contactar con servicio técnico. En la figura 49, se puede observar el lugar de la interfaz donde se seleccionan.


*Figura 49. Selecciona Célula.*

Para el archivo Excel, figura 50, se debe clicar en la carpeta. Después saldrá una ventana emergente, dónde deberá buscar un archivo Excel **YA CREADO**, figura 51.


*Figura 50. Abre directorio.*

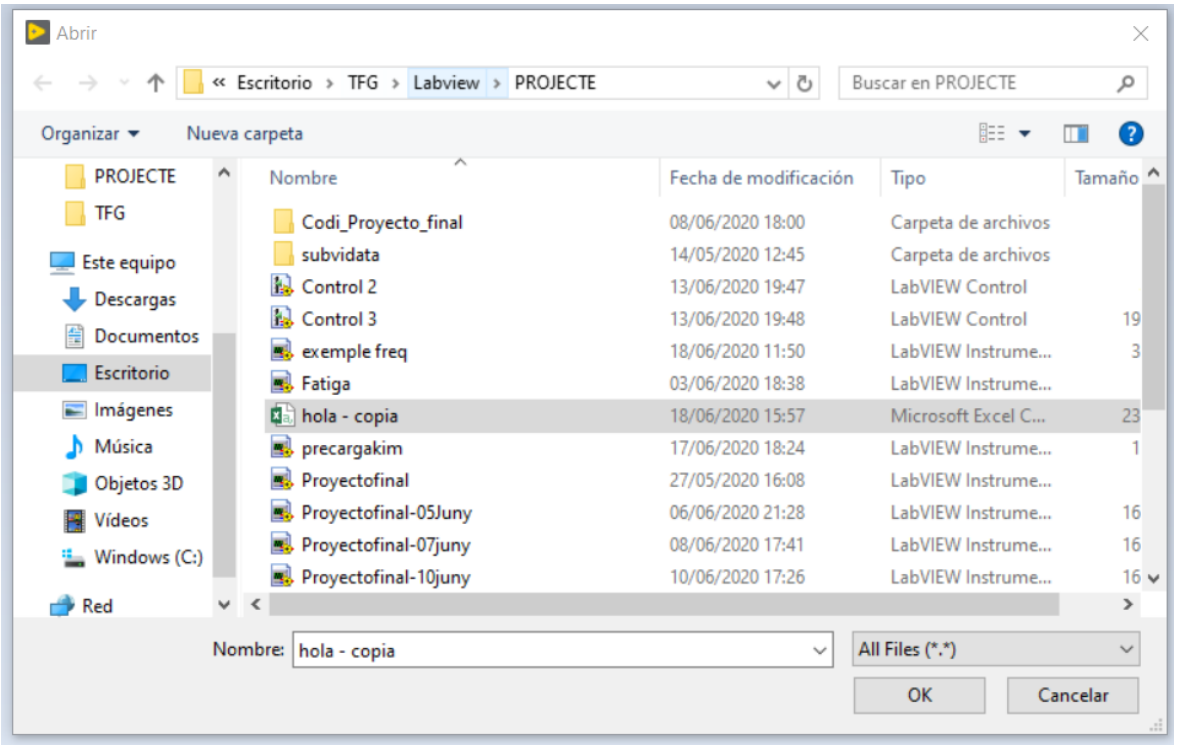



*Figura 51.Seleccionar archivo Excel.*

## Selección tipo de ensayo.

En la misma pantalla de interfaz podemos observar a la derecha tipos de ensayos, Figura 52.

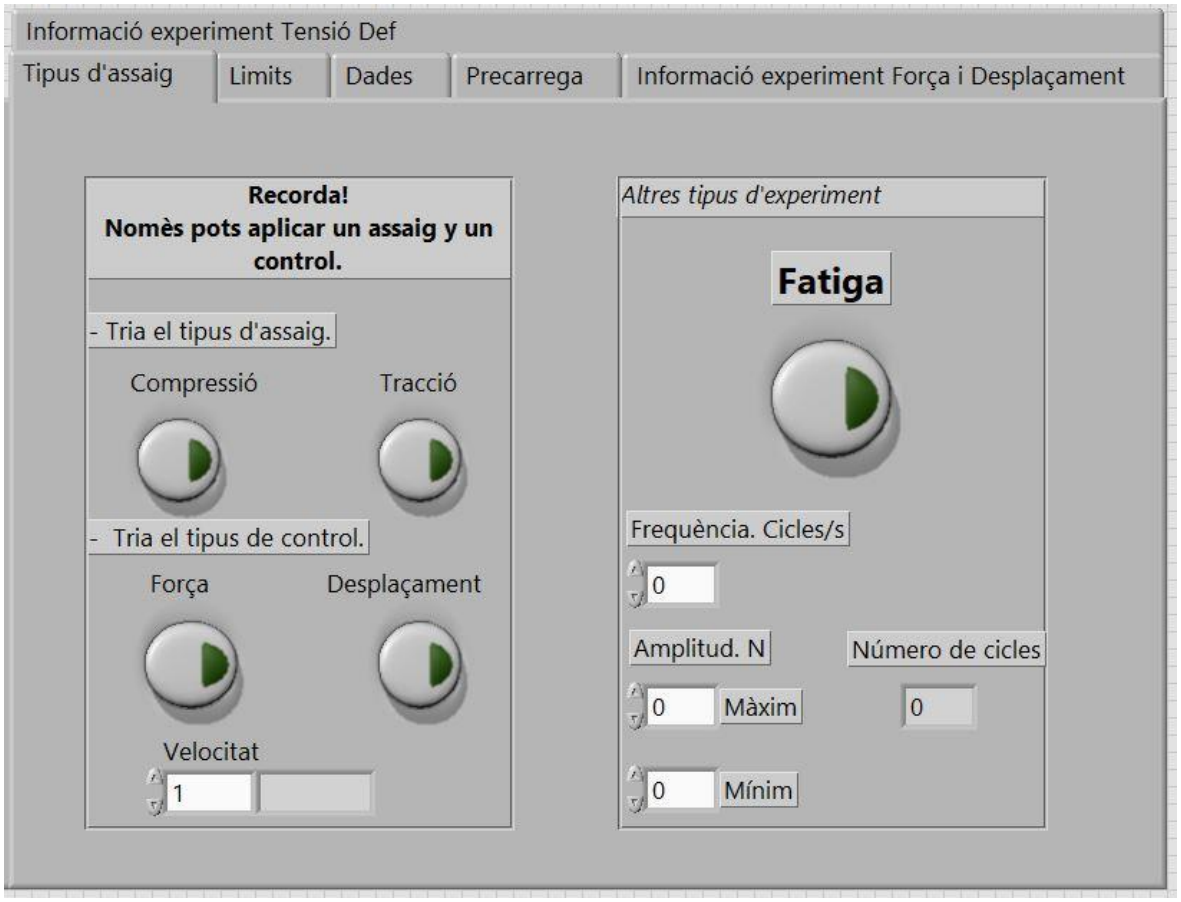



*Figura 52. Selección tipo de ensayo.*

Aquí se debe seleccionar que tipo de ensayo se quiere si tracción/compresión y fuerza/desplazamiento, únicamente se debe clicar encender uno de cada tipo. Además, deberá rellenar el valor de velocidad deseado, en un tipo de control de Fuerza será en Newtons y uno de desplazamiento en mm/min.

En el caso que se desee realizar fatiga se debe omitir el anterior párrafo, y se deberá clicar el botón de Fatiga, y rellenar los valores de Frecuencia, Amplitud y de Número de ciclos.

## Límites.

Para colocar los límites se deberá navegar en las pestañas del bloque derecho y presionar límites. En la figura 53 se observan cómo.

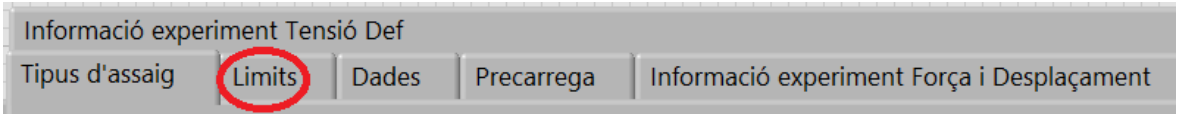



*Figura 53. Selección Límite.*

Posteriormente se deberán rellenar los valores de límite de desplazamiento y de carga, figura 54.


*Figura 54. Rellenar límites.*

Si observa que un Led está en rojo, como en la imagen "Carrega Inferior", este quiere decir que el límite se ha traspasado. El programa únicamente será de lectura. Los demás fallos se explicarán el TroubleShooting.

## Datos probeta.

Antes de empezar el ensayo se deberán indicar las dimensiones de la probeta. Esto servirá para poder observar los valores de Tensión y deformación. Estas se encuentran en una pestaña junto a límites, se puede observar en la figura 55. Para facilitar el rellenado de los datos, dispone de una fotografía que indica que se pide en cada espacio.

Si no se dispone de extensómetro, esta medida no dejará de ser una mera aproximación.

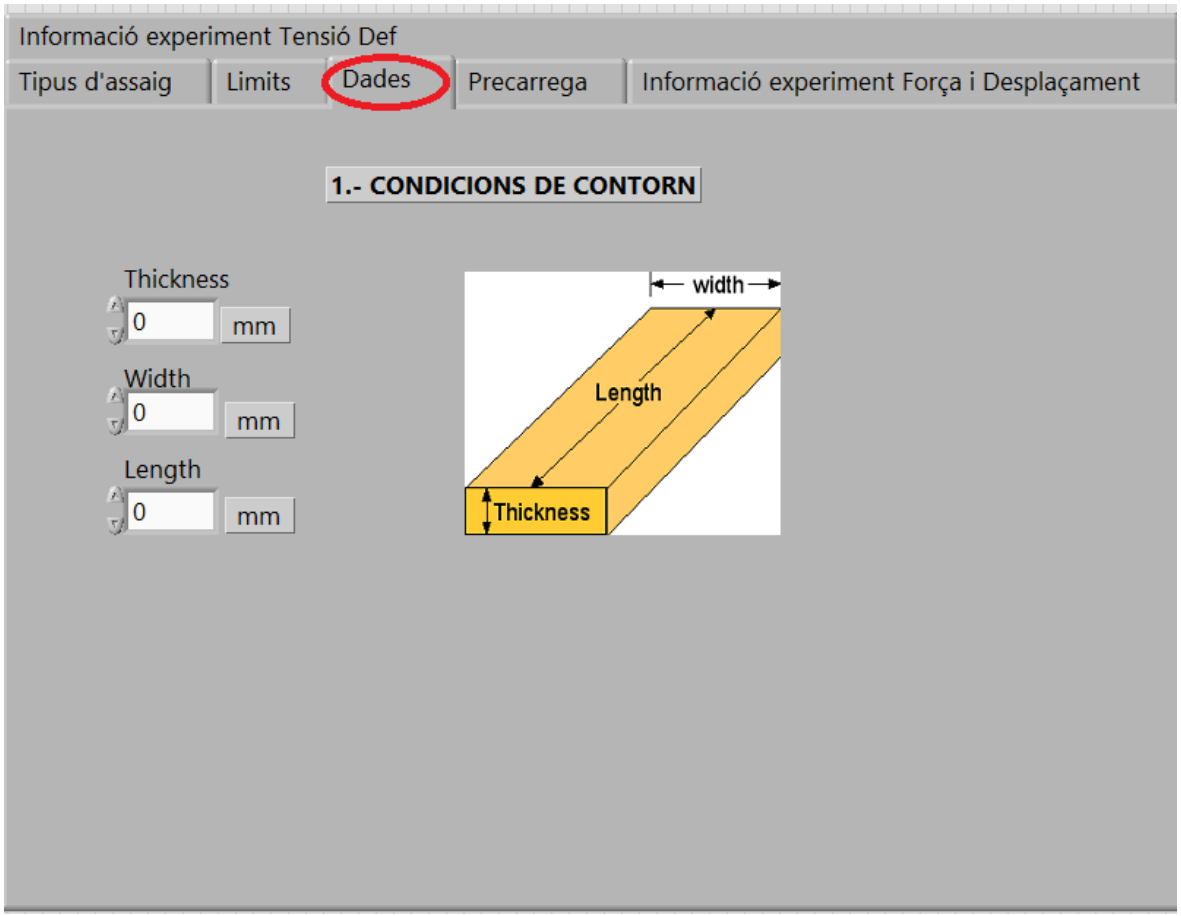



*Figura 55. Datos probeta.*

## Precarga.

En el caso que quiera hacer precarga, deberá navegar otra vez por las pestañas de Límites, Datos, etc.

Ahí se encuentra Precarga. En Precarga se dispone de un pulsador, figura 56, un valor de Precarga que deberá rellenar, y un led que indica si la precarga ha sido aplicada correctamente. En el caso que no desee precarga ignore este punto.

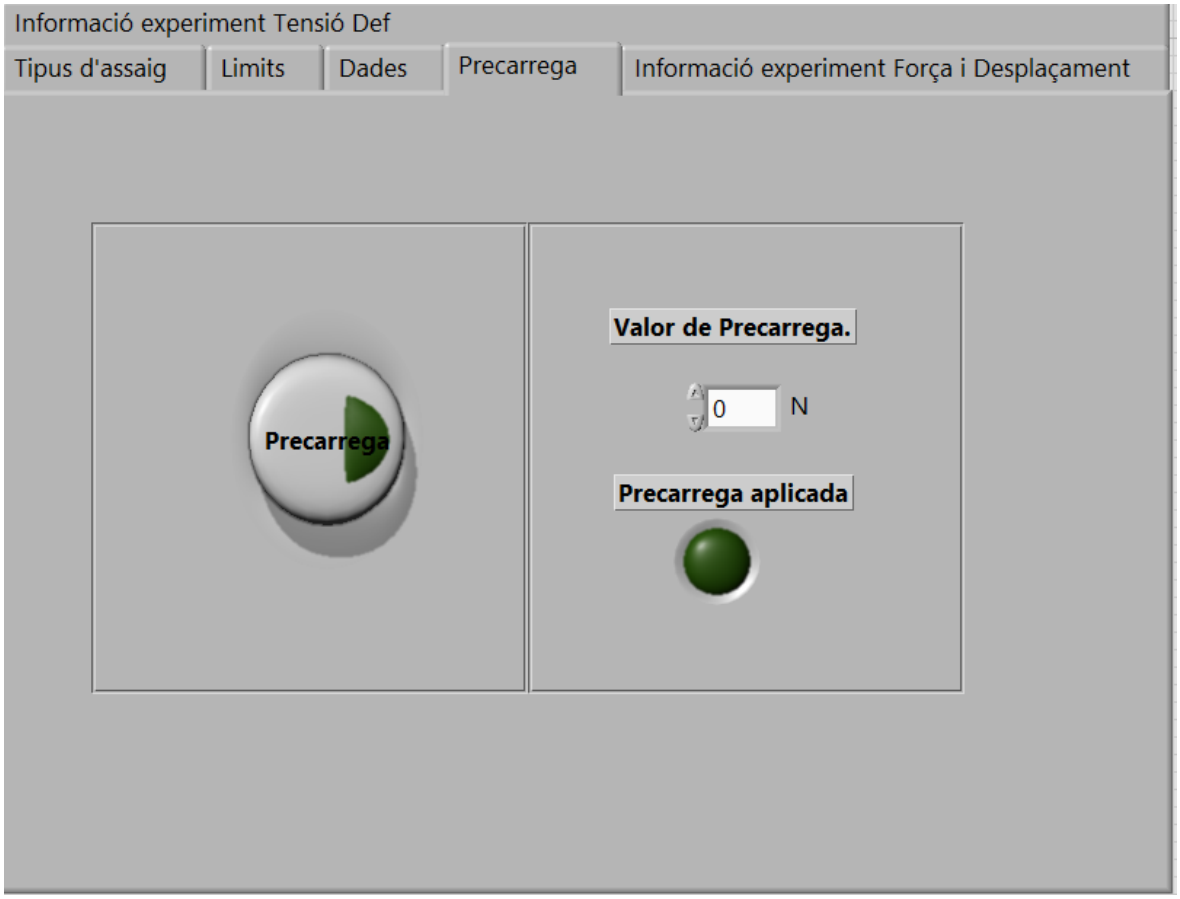



*Figura 56. Pestaña Precarga.*

## Antes de empezar el ensayo.

Una vez colocado todos los condicionantes se deberá clicar Start. Antes, puede familiarizarse con el entorno. Arriba se puede observar el tiempo existente que lleva el ensayo en marcha. También tiene la posibilidad de realizar una Tara al desplazamiento y la fuerza si desea empezar el ensayo con ambos valores en cero. En la Figura 57, se pueden observar ambos.

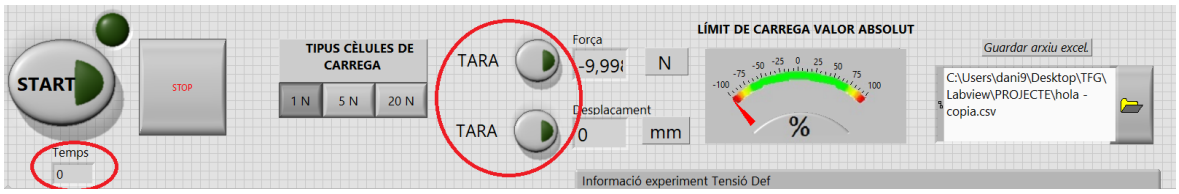



*Figura 57. Tara y tiempo.*

También dispone de un gráfico que mide el límite de carga de la célula de carga, figura 58. Este mide el tanto por ciento de capacidad que tiene la célula de carga.


*Figura 58. Límite de Carga.*

## Start y Stop.

Para empezar el ensayo se deberá clicar el botón Start, a su vez que un botón Stop que permite un paro manual de la máquina. Una vez clicado Start, dispone de diferentes informaciones.

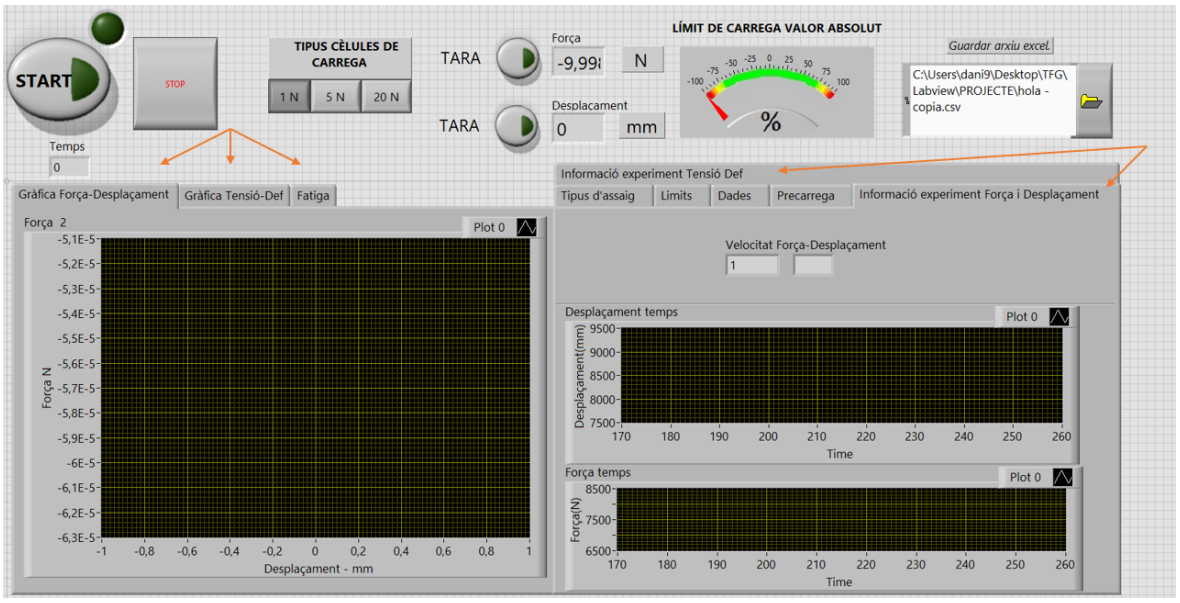



*Figura 59.Información ensayo.*

En la figura 59, se puede observar las diferentes informaciones que se pueden leer durante el ensayo. En las pestañas de la izquierda se encuentran las gráficas Fuerza vs. Desplazamiento, Tensión vs. Deformación y de Fatiga (Sinusoidal Fuerza vs. tiempo). Además, en la pestaña de la derecha encontramos información para los dos primeros, donde se observan la velocidad del ensayo, y los gráficos respecto el tiempo.

# Troubleshooting.

Este apartado consiste en ofrecer soluciones a futuros problemas que puedan ocurrir con el Software. Antes de acudir a buscar el problema, se deben revisar que los pulsadores estén bien conectados y que no haya un error entrando los datos.

| Problemas con la célula de carga | |
|---|---|
| No se lee correctamente la carga. | Se debe comprobar la conexión del amplificador con el Arduino. Y la selección de célula en LabView.<br><br>Si el error persiste intente recalibrar la máquina. |
| La lectura de la célula de carga no varía. | Se debe comprobar las conexiones de la célula de carga. |
| No lee la célula de carga. | Compruebe las conexiones de la célula de carga. Si el problema persiste, comprueba que la célula de carga funcione midiendo con un multímetro los valores de resistencia. |
| Motor. | |
| El motor no avanza ni retrocede pero está en movimiento. | Compruebe que los pins estén conectados correctamente en el Arduino, y las conexiones de voltaje. |

# 6. Esquema de uso.

1. Seleccione célula de carga y el archivo Excel.

2. Seleccione el tipo de ensayo o Fatiga y el tipo de control. Asigne una velocidad.

3. Coloque los límites correspondientes.

4. Introduzca las dimensiones de la probeta.

5. Coloque los valores de PID sugeridos.

6. Aplique una precarga si es necesario.

7. Tara si es necesario.

8. Comience cuando deseé mediante el pulsador Start.

9. Utilice la interfaz que sea más cómoda para su ensayo.

10. Una vez finalizado, pare el programa mediante Stop.

# 7. Conclusiones

Se ha creado una máquina de ensayo universal “casera”, con un software nuevo y moderno que permite realizar nuevos ensayos al campus.

Debido a la situación vivida el cuadrimestre de primavera del 2020, el trabajo ha estado más centrado en la creación del software que en mejorar la aplicación de la máquina, debido a la imposibilidad de mejorarla sin un taller y herramientas necesarias. Esto ha causado que distintos problemas de fallos de electrónica, implicaron un gasto mayor en recursos y pérdida de tiempo en realizar el proyecto, por lo que el software podría haber abarcado más.

El concepto de reingeniería cada vez es más importante hoy en día y queda demostrado con este proyecto que puede resultar muy útil, económico y enriquecedor para el ingeniero o futuro ingeniero. Reduce enormemente los costes, ya que con máquinas no funcionales se pueden conseguir máquinas de última tecnología. Enriquecedor debido a que aprender el funcionamiento de las máquinas que nos rodean cada día puede ayudar en mejorar su mantenimiento, a interpretar los datos y a optimizar el su funcionamiento. También este tipo de proyectos enriquece al estudiante debido a la gran cantidad de información y formación que debe digerir, adquiriendo conocimientos de gestión e ingeniería de proyectos, automática, señales y electrónica además de la programación.

Se ha observado que mediante el software utilizado se puede realizar cualquier tipo de máquina de laboratorio. Incluso se puede utilizar para expandir funciones de una máquina del laboratorio.

Este proyecto no está cerrado ya que, da pie a que se pueda optimizar aún más el software propuesto, o incluso a aplicar reingeniería a otras máquinas de la facultad.

# 8. Presupuesto y/o Análisis Económico

En este apartado se va a realizar el presupuesto de este proyecto y se va a comparar con la adquisición de una máquina universal de ensayos en el mercado.

## 8.1. Coste de material.

| Producto | Cantidad | Coste/u.a | Coste total € |
|---|---|---|---|
| **Placa Arduino Mega** | 1 | 13€/ud | 13€ |
| **Cable USB Arduino** | 1 | 0,5€/ud | 0,5€ |
| **Célula de carga 5N** | 1 | 8,19€/ud | 8,19€ |
| **Célula de Carga 20N con HX711** | 1 | 1,72€/ud | 1,72€ |
| **Husillo 30mm con tuercas** | 2 | 11€/ud | 22€ |
| **Poleas GT2, 16T** | 1 | 1,17€/ud | 1,17€ |
| **Poleas GT2, 30T** | 1 | 2,29€/ud | 2,29€ |
| **Poleas GT2, 60T** | 1 | 1,09€/ud | 1,09€ |
| **Correa de distribución** | 1 | 3,48€/ud | 3,48€ |
| **VMA 409** | 1 | 2,53€/ud | 2,53€ |
| **Rodamientos MR126ZZ** | 10 | 0,74€/ud | 7,4€ |

| | | | |
|---|---|---|---|
| **Casquillos autolubricantes BQLZR** | 2 | 6,92€/ud | 13,84€ |
| **Rodamientos F6-12M** | 1 | 2,38€/ud | 2,38€ |
| **Motor tipo Mabuchi 130** | 1 | 2,95€/ud | 2,95€ |
| **Motor reductor 6v. Ref.156128B2** | 1 | 12,04€/ud | 12,04€ |
| **Encoder VMA435** | 1 | 3,79€/ud | 3,79€ |
| **Acero A36** | 10 kg | 6,79€/kg | 67,9€ |
| **Columnas de Acero R=2mm L=45mm** | 2 | 9,7€/ud | 19,4€ |
| **Placa Aluminio** | 3,34 Kg | 0,7€/kg | 2,34€ |
| **Kit Electrónica, resistencias y cables** | 1 | 3,94€/ud | 3,94€ |
| | | | |

**Coste de los materiales 192€.**

## 8.2 Coste personal involucrado.

La creación de la máquina costó alrededor de 62h por operario. Se necesitaron 2 operarios para completarla.

| **Personal** | **Cantidad(h)** | **Coste(€/h)** | **Coste total(€)** |
|---|---|---|---|
| **Proyectista** | 62 | 10 | 620 |

| | | | |
|---|---|---|---|
| **Profesor** | 62 | 60 | 3720 |

Para la creación del Software:

| **Personal** | **Cantidad(h)** | **Coste(€/h)** | **Coste total(€)** |
|---|---|---|---|
| **Proyectista** | 332,5 | 10 | 3325 |
| **Profesor** | 20 | 60 | 1200 |

**Total 8.865€**

## 8.3 Resumen análisis económico.

El coste total de la creación de la máquina y del software asciende a los 9.057€.

# Bibliografía

[1] William D. Calister, Jr.: "Introducción a la Ciencia e Ingeniería de Materiales 1". Editorial Reverte.

[2]https://www.researchgate.net/publication/321356419_Ensayo_de_traccion_segun_norma_DIN_53455#:~:text=Según%20norma%2C%20el%20ensayo%20de,colombiana%20con%20fines%20no%20comerciales. Visitado 01-06-2020.

[3] http://www.ni.com/tutorial/7138/es/ - Visitado 4-06-2020.

[4] https://aprendiendoarduino.wordpress.com/tag/motor-paso-a-paso/#:~:text=Motor%20Paso%20a%20Paso%20con%20Arduino&text=Un%20motor%20paso%20a%20paso,%C3%A9l%20en%20la%20secuencia%20correcta. Visitado 4-06-2020.

[5] http://www1.frm.utn.edu.ar/mielectricas/docs2/PaP/MOTOR_PaP_FINAL.pdf. –Facultad Regional de Mendoza, Argentina. Visitado 4-06-2020.

[6] https://www.luisllamas.es/motores-paso-paso-arduino-driver-a4988-drv8825/ - Visitado 6-06-2020.

[7] Agredo, Martha Natalia, Quintana, Julián, & Flórez, Juan Fernando. (2015). *Diseño y pruebas de un sistema de monitoreo y supervisión para una máquina universal de ensayos. Prospectiva*, *13*(2), 25-37.

[8] J. A. Redondo, *Reparación y automatización de la máquina universal de ensayos Acco Riehle de la UNET,* Universidad Nacional Experimental de Táchira, 2009.

[9] https://aprendiendoarduino.wordpress.com/2016/11/16/librerias-arduino-2/ Visitado 6-06-2020.

[10] P.Gallego, R. Claros, *Diseño mecánico de una máquina universal de ensayos para polímeros.* Escuelda de tecnología mecánica, Pereira-Risaralda, 2009.

[11] Jordi Mayné "Sensores Acondicionadores y Procesadores de Señal" Silica, Rev2. 2003.

[12] Orellana García José Israel y, Tello Salazar David Hector. *Medición de esfuerzos y deformaciones en barras metálicas utilizando galgas extensométricas*. Tesis de grado - FIEC.

[13] http://www.valvehydraulic.info/creation-and-control-of-fluid-flow/hydraulic-servo-valves.html Visitado 19-06-2020

[14] https://www.youtube.com/watch?v=Y4j_uGRPYes Vistado 19-06-2020

[15] https://www.arduino.cc/en/tutorial/PWM Vistado 19-06-2020

[16] https://learn.sparkfun.com/tutorials/pulse-width-modulation/all Vistado 19-06-2020

[17] https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019OkFSAU&l=es-ES Vistado 19-06-2020

[18] J.Redrejo, *Desarrollo de sistemas de regulación y control.* IES Santiago Apostol.

# Anexo

## Programas obsoletos.

La creación de un software de cero provoca que las ideas que se tienen de cómo se va a realizar vayan cambiando, por motivos de ahorrar espacio, de simplificar diagramas o de evitar errores. Por eso, se encuentra interesante comentar programas que se iban a colocar, porque se hacían y porqué se han cambiado.

**Precarga**

Para el programa de precarga sucedió varias veces y los motivos fueron los siguientes:

1. Se quería ahorrar espacio, por lo que se pensó que realizar un subprograma sería más interesante.
2. Se quería realizar antes de empezar el programa, incluso antes de llegar a definir las condiciones del ensayo.
3. Para aplicarla se debían realizar cambios en las variables principales, como son el sentido y la velocidad, y para evitar complicar el código se deseaba realizar aparte.

Por qué al final no se realizó:

1. El bucle que contenía el subprograma era casi idéntico al programa principal, por lo que controlando la navegación entre casos se podía conseguir reducir la memoria que consumía, haciendo el programa de errores.
2. Realizar bucles dentro de subprograma es una fuente de errores, es decir inducen a que el programa se pueda quedar estancado.
3. Se deseaba que la máquina realizará movimientos mediante los pulsadores antes que realizar la precarga, cosa que el subprograma no permitía.
4. Además, lo más importante, el programa principal controla los límites y la capacidad de la célula de carga, esto evita que pueda romper partes de la máquina como es la célula de carga. Esta característica no la poseía el programa Precarga, por lo que el usuario podía aplicar una precarga superior a la capacidad de la célula de carga colocada.

En conclusión, gracias a este subprograma se han realizado las mismas funciones dentro del programa, por lo que ha ayudado a aclarar el proceso de precarga. Además, se han podido contemplar errores y fallos que no estaban en mente.

**Programa obsoleto Precarga.**

La precarga es una fuerza que se hace con anterioridad al ensayo por distintos motivos. Por ejemplo, en ensayo a tracción se suele hacer para observar si la probeta está bien colocada con las mordazas, para tensar la probeta y que tenga un cero correcto en la gráfica entre otros. También en ensayos de fatiga se recomienda realizar una precarga al máximo de amplitud, por ejemplo, si ensayamos de 500N a -500N, se aplica una precarga de 500N al empezar, así se empieza en un pico de amplitud.

En este proyecto, se dará al usuario la posibilidad de aplicar una precarga. Para ello deberá acceder a una pestaña en el bloc de definición del experimento, dar un valor de precarga y clicar en el pulsador. Esta precarga se realizará una vez el programa se encienda y se aplique el botón Start. Además, cuando esta precarga se haya aplicado una ventana emergente notificará que se ha aplicado correctamente.

La explicación del programa puede ser un poco difícil para aquel que no esté familiarizado con la interfaz de LabView por lo que se intentará explicar cada elemento que se enseñe.

El código principal se muestra en la figura 60.

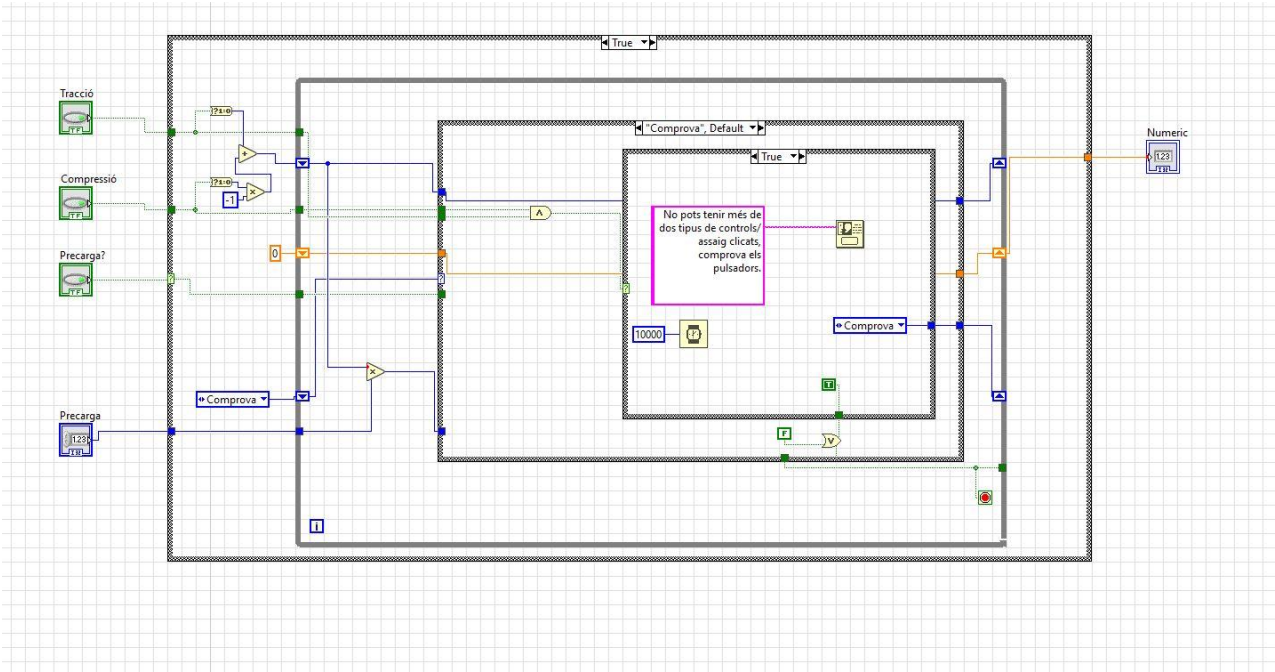



*Figura 60. Programa precarga inicio.*

Una vez se abre el programa de precarga se observa el siguiente código de bloque. A la izquierda tenemos los inputs que el programa necesita, que el programa principal proporcionara. A la derecha en cambio tenemos los outputs que el programa recibirá y mostrará al usuario. Para los inputs tenemos, los pulsadores de Tracción, Compresión, ¿Precarga? y Precarga. Tracción y compresión son los pulsadores que el programa necesita para arrancar, para saber en cuál sentido debe girar el motor. También tenemos el botón de Precarga?, este será pulsado cuando el usuario desee realizar precarga,

por lo contrario, el programa principal no realizará nada de este programa y únicamente saltará a otra parte del código. Por último, tenemos Precarga en azul, el color azul significa numérico, por lo que este es el valor exacto en Newtons de precarga que se necesita. Pasando la primera barrera tenemos un Case Structure, este tipo de estructuras sirven para realizar diferente código según un valor, por ejemplo, el Case Structure está determinado por el valor del pulsador de Precarga?, si este esta pulsado, el valor será True o encendido, por lo que el programa entrara en el caso Encendido, en cambio, si el botón está apagado se mostrará el caso False, mostrado en la Figura 61.

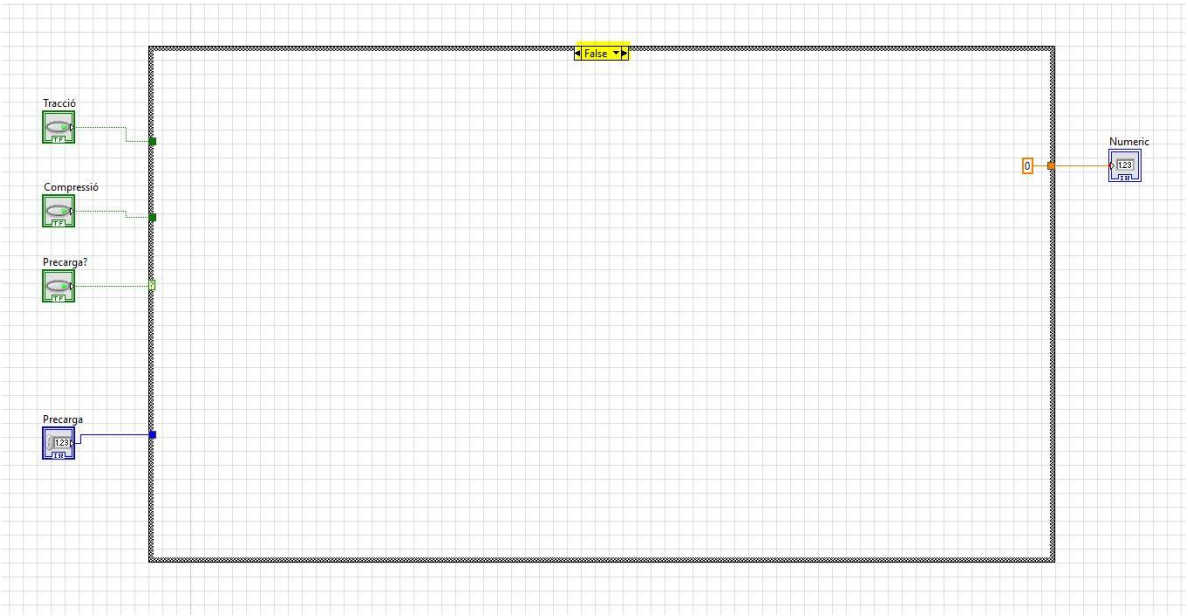



*Figura 61. Precarga False.*

En este caso se le dará al output un valor 0 y acabará el programa de precarga.

Volviendo a la figura 62, entramos en el Case Structure, ahí tenemos unos multiplicadores y un 0,1 bolean. Estos tienen la función de transformar el valor True y False en 1 o 0 a en función de si están los botones de tracción y compresión encendidos o apagados.

También observamos la palabra Comprova. Para navegar en diferentes partes de todos los programas realizados se ha optado por una organización tipo Case Structure. Como hemos dicho anteriormente, en el Case Structure se mostrará el código que el usuario desea utilizar, en cambio el que no se desea se omite. Además de valores True o False, este tipo de estructura permite tener casos de Lista llamados Enum en LabView, esto da la posibilidad al programador de realizar un "Path", en inglés un camino, por el que el código debe pasar siempre que cumpla unas condiciones. Más adelante se entenderá al ver los ejemplos.

Al pasar la siguiente pared observamos otro tipo de estructura, a este tipo se le llama While. Se puede observar gráficamente una diferencia, esta es de un color gris oscuro cuando la otra tiene filigranas de color negro con el fondo blanco, además, funcionan diferente. Este tipo de estructuras crean un bucle hasta que reciba un Stop. El Stop puede ser por lo general de tres maneras, el primero suele ser a que existe un controlador que cuando este esté pulsado se pare el programa, otra manera es con el botón abort arriba a la derecha de la interfaz de LabView, o cabe la posibilidad de que una condición habilite el Stop, es decir, que el programador por ejemplo diga cuando x sea más grande que y, Stop.

Este tipo de estructuras permiten Shift Registrer se pueden observar con una forma de flecha en ambos lados de la estructura While. Este recurso sirve para guardar valores de un bucle a otro, por ejemplo, si en cada bucle queremos sumar 1, realizando un Shift Registrer con una operación de suma más uno y conectando el resultado al Shift Registrer opuesto, obtendremos para cada iteración un +1.

Después del bucle While tenemos otro Structure Case, este corresponde al tipo que contiene una lista, en este caso la lista es la siguiente:

1- Comprova. Figura 62.
2- Mou. Figura 63.
3- Compara. Figura 64.
4- És més petit. Figura 65.
5- És més gran. Figura 66.
6- Acaba. Figura 67.

**Comprova.**

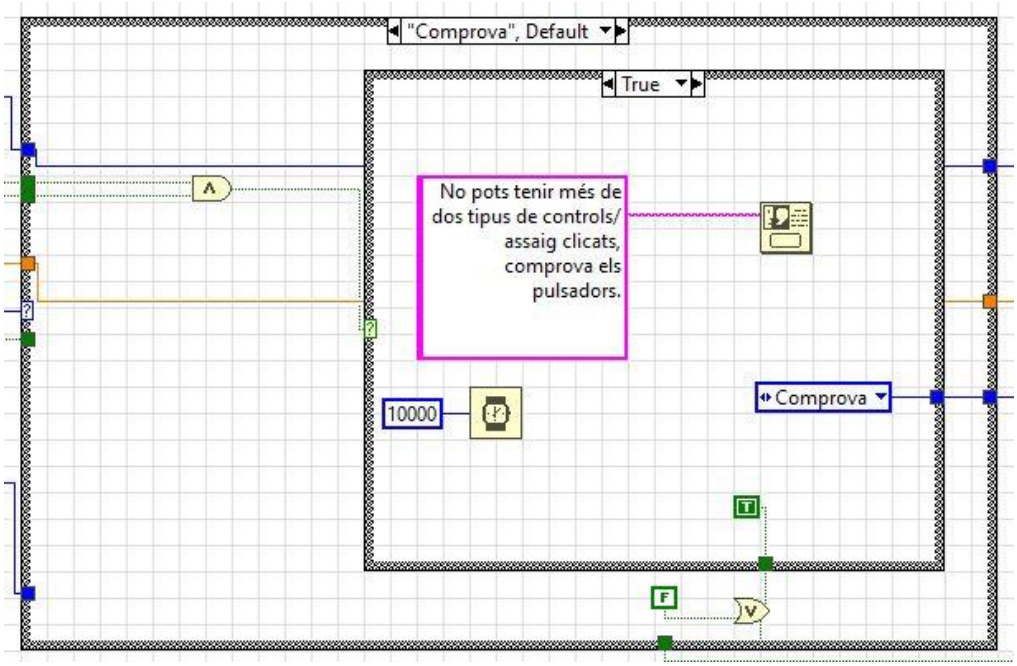



*Figura 62. Caso comprova.*

Este caso comprueba que los botones estén pulsados correctamente, es decir, sería una incoherencia poder realizar el ensayo con tracción y compresión pulsados, por ende, debemos limitarlo de esta manera. Al estar los dos pulsados, LabView emitirá una ventana emergente anunciando que no se pueden tener más de dos pulsadores clicados, y se esperará 10 segundos hasta que se dejen de pulsar, si en este tiempo no ha habido ningún cambio, seguirá emitiendo una ventana emergente. En el caso de que los pulsadores estén correctos se avanzará al siguiente en la lista Mou.

## Mou.

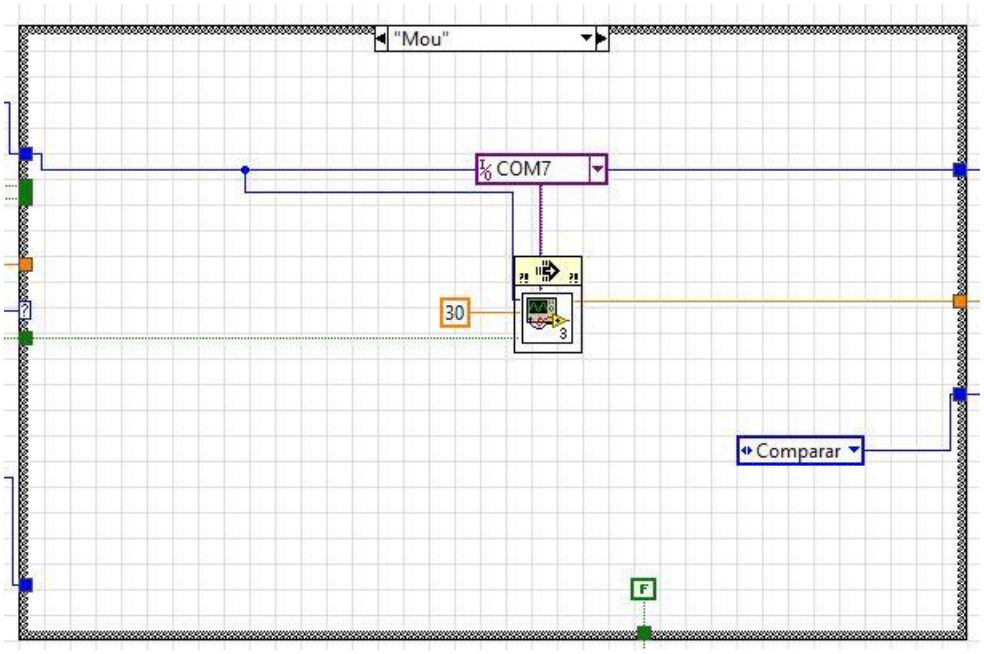



*Figura 63. Precarga Mou.*

Este caso moverá el motor mediante el programa ya realizado SubVi Data, en el sentido indicado y enviará el valor de la célula de carga al programa para que opere con él. Después pasará al caso Comparar.

## Comparar.

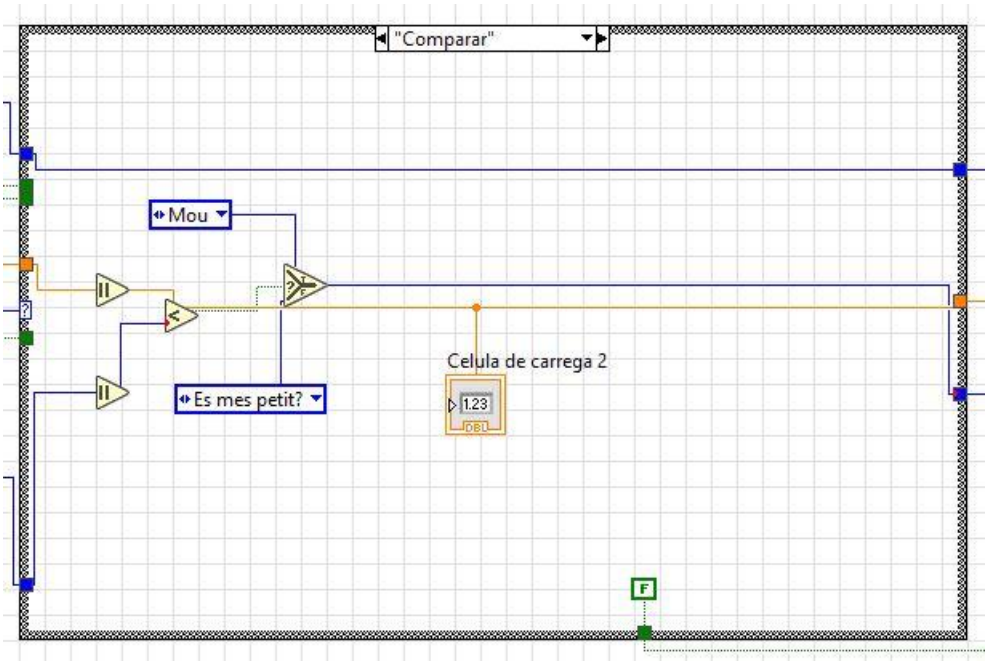



*Figura 64. Precarga comparar.*

Al principio de este programa nos encontramos un operador repetido que no se ha mostrado anteriormente, este operador devuelve el valor absoluto del número recibido. Por lo que valores de tracción o compresión no modificarán el procedimiento. Después nos encontramos con un selector, este selector hará moverse a diferentes casos según si el valor de la célula de carga es más pequeño o es más grande que el de la precarga a aplicar. En el caso de que sea más pequeño, volverá al caso Mou, por lo que enviará a Arduino que debe moverse aún más. En cambio, si detecta que el valor es más pequeño, es decir que la célula de carga supera el valor de precarga vaya al siguiente caso És més petit?

## És més petit?

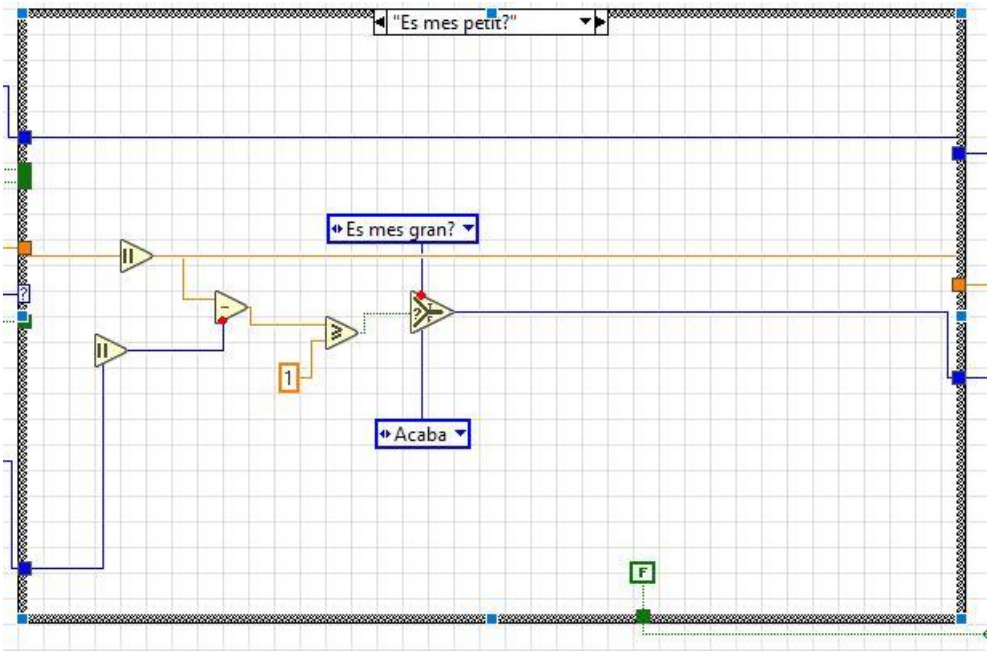



*Figura 65. Precarga És més petit?*

En este caso se realiza una suposición, primero de todo resta el valor absoluto de la precarga a desear y la célula de carga. Este resultado debe ser ±1N, admitiendo que tenga un error. En el caso de que el valor sea mayor a 1, se irá a És més Gran? En cambio, si cumple el Error irá a Acaba.

## És més gran?

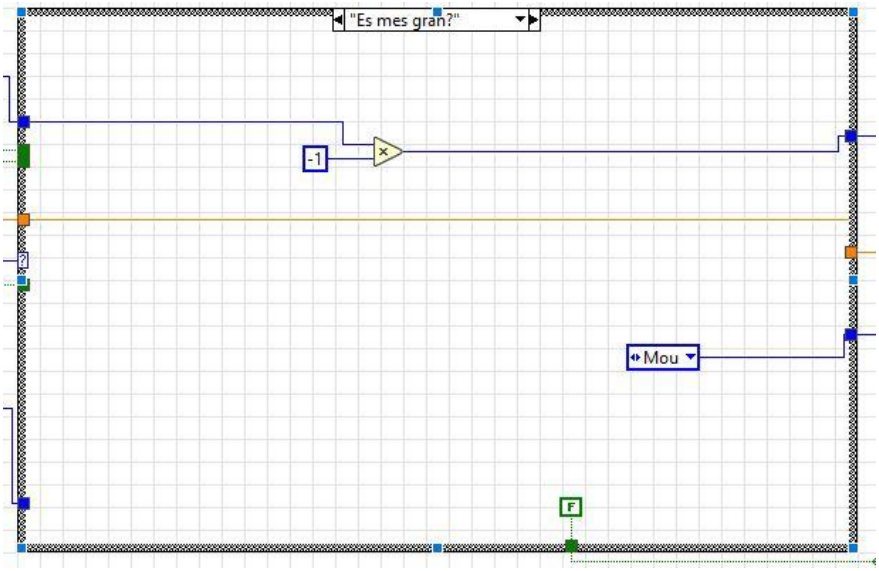



*Figura 66. Precarga és més gran?*

Este caso se realiza por la posibilidad que el valor de la célula de carga haya pasado el valor deseado de precarga, eso provocará que el programa se parará sin haberla aplicado correctamente. Para arreglar este posible error, se realiza este caso, hace que el motor, dé marcha atrás, es decir, que, si está girando en el sentido de la tracción, cambie de giro a compresión hasta que encuentre el valor correcto. Si otra vez pasará el mismo error, este caso también haría cambiar el sentido de nuevo.

**Acaba.**

Por último, si en És més petit? El valor está dentro del error se moverá a este caso.

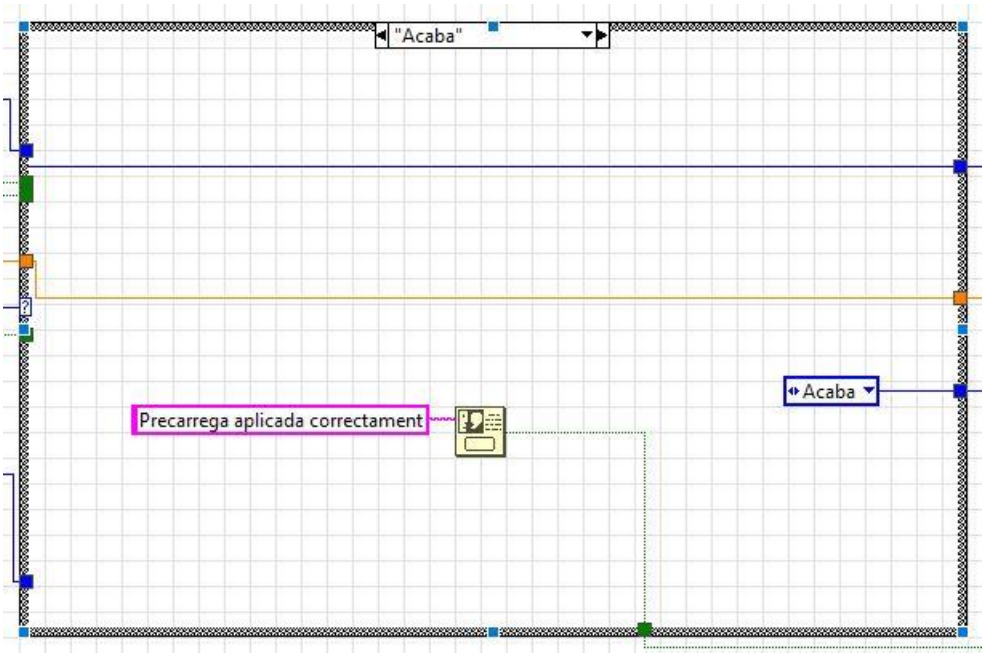



*Figura 67. Precarga Acaba.*

Este caso únicamente ejecutará una notificación emergente de que la Precarga esta aplicada correctamente y le dará un valor True al Stop del While que se ha hablado anteriormente, este True hará que el programa finalice.

## Esquemas eléctricos

Para el control y funcionamiento de la máquina de ensayo universal, se han ensamblado los siguientes circuitos.

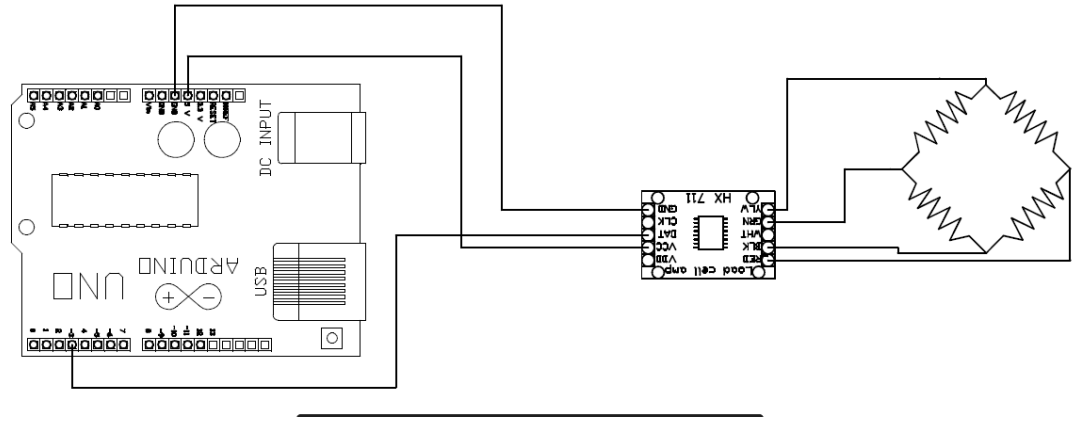



*Figura 68. Esquema eléctrico célula de carga.*

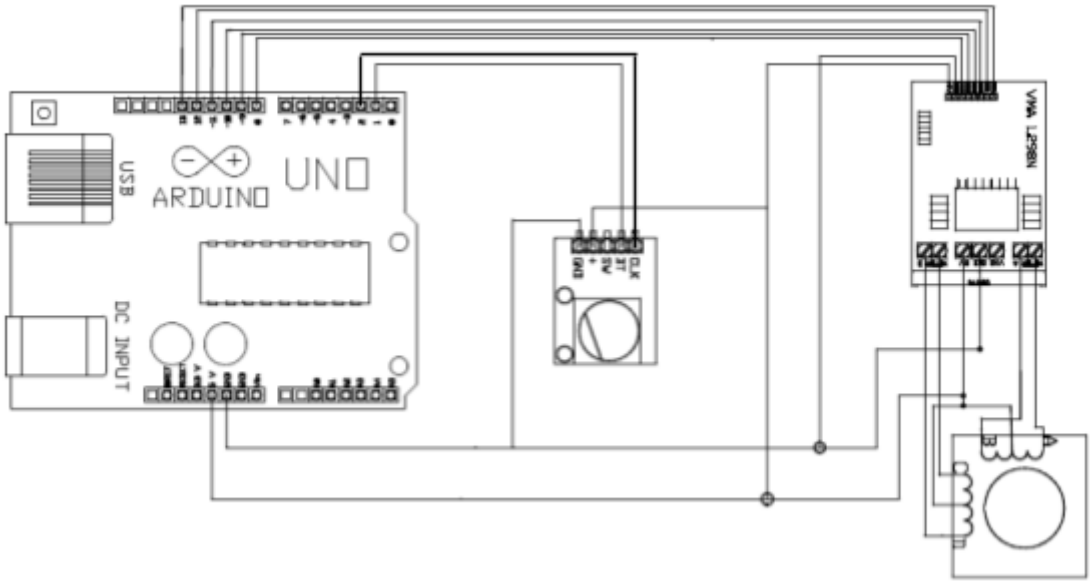



*Figura 69. Esquema eléctrico motor.*

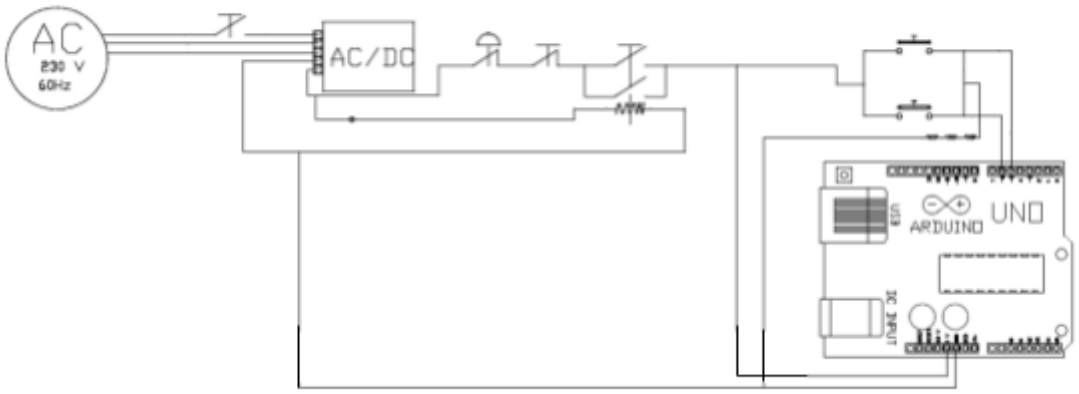



*Figura 70. Esquema eléctrico fuente de alimentación, pulsadores y seta.*

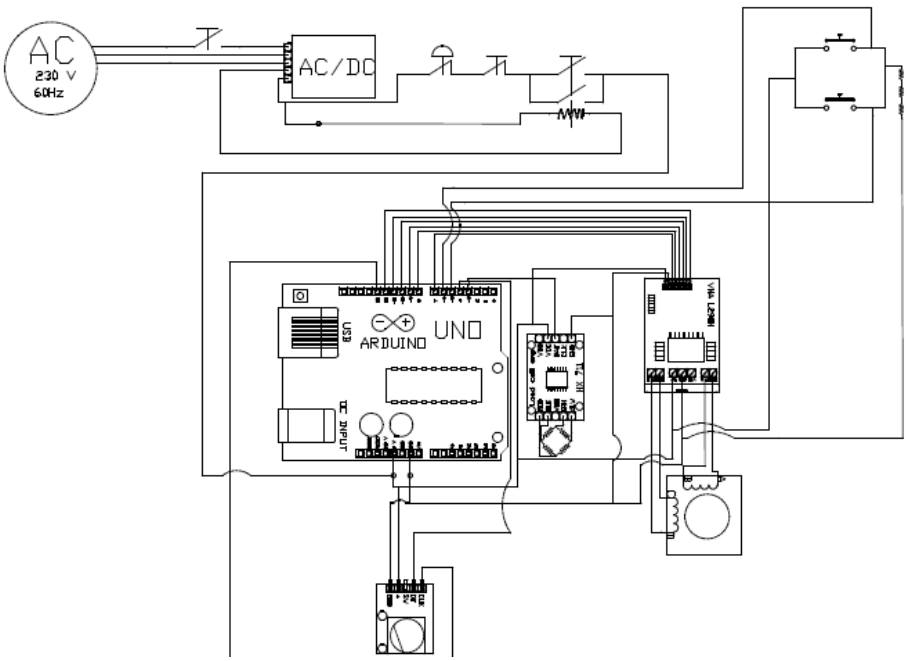



*Figura 71. Esquema eléctrico general.*